PROJECTE DE LLIURE ELECCIÓ ANDROID

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Manel Simon i Martínez
i dirigit per
Ramón Grau Sala
Bellaterra, 17 de Juny de 2015

Universitat Autònoma de Barcelona

escola d'enginyeria

El sotasignat, Ramon Grau Sala

professor/a de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en/na Manel Simon Martínez

I per a que consti firma la present.

Signat:

Bellaterra, 16 de Juny de 2015

Taula de Continguts

INTRODUCCIÓ

Un telèfon intel·ligent és aquell capaç de combinar les funcionalitats bàsiques de telefonia amb aquelles pròpies d'un ordinador: la computació. Aquest concepte va ser postulat per Theodore Paraskevakos en 1971 i patentat en 1974, però no va ser fins el 1993 quan es varen comercialitzar les primeres unitats.

Durant l'últim tram de la dècada dels 90, diversos prototips i productes finals van anar apareixent de la mà de grans companyies com IBM, Ericsson o Nokia, on cada cop s'assolien nous reptes com la integració de teclats físics QWERTY, aplicacions de gestió per correu electrònic o la possibilitat de navegar per la Internet.

Tenint en compte les empreses que donaven suport a aquests dispositius durant els últims anys del segle XX, i les empreses que en un futur no molt llunyà ho farien (com Apple o Google), no és sorprenent que a dia d'avui, l'any 2015, aquests telèfons intel·ligents hagin assolit una maduresa extraordinària i s'hagin convertit en un element indispensable per la vida quotidiana de gairebé tothom. De fet, la irrupció de la Internet ha fet que els anomenats *smatphones* cada cop deixin més de banda la part de telefonia i es centrin més en les funcionalitats pròpies d'un ordinador: execució d'aplicacions amb interfície gràfica, gestió eficient i òptima de les comunicacions i gran capacitat de computació.

Tal és la penetració d'aquests dispositius en la vida normal de les persones que s'estima que a l'any 2017 hi haurà 5'13 bilions de telèfons intel·ligents al món, pertanyents a un 69% de la població mundial [1]. I així aquests dispositius passen de ser una mera innovació tecnològica a ser una potent i idíl·lica plataforma per a la irrupció de models de negoci innovadors, associats a les aplicacions mòbils i a la potència de comunicació que ofereix Internet.

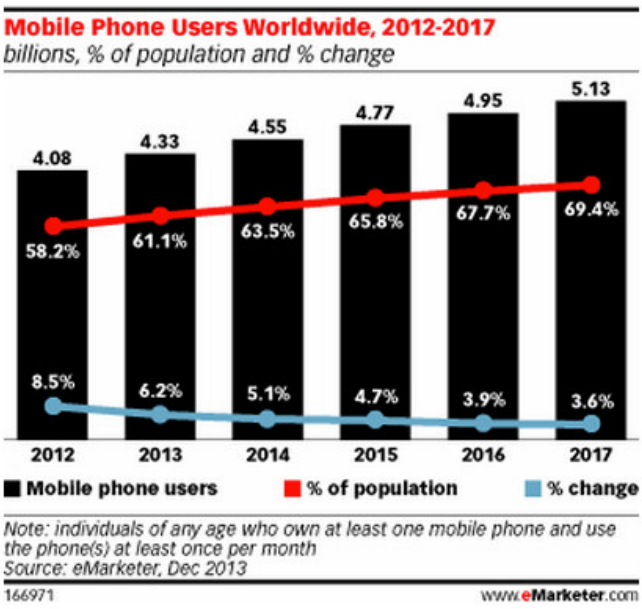


Figura 1. Penetració mundial d'smpartphones

En l'actualitat, Apple i Google són les principals companyies que implementen sistemes operatius per a dispositius mòbils, catalogant al voltant de 1.250 i 1.500 milions d'aplicacions disponibles per a les respectives plataformes. A més a més de la rapidesa amb la que aquests dispositius evolucionen, val a dir que els models de negoci innovadors que hem esmentat no s'han quedat enrere.

Primerament, el model més popular va ser la venta directa d'aplicacions a través de les tendes que oferien ambdues empreses: Apple Store i Google Play. L'estratègia era oferir una versió de prova de forma gratuïta de l'aplicació per convèncer a l'usuari de l'adquisició de la versió completa. Poc a poc va anar apareixent el model de negoci basat en la publicitat, és a dir, les aplicacions van passar a ser gratuïtes però mostraven anuncis gràfics durant l'execució, amb la qual cosa els desenvolupadors guanyaven ingressos per part de les empreses que es volien anunciar.

Els darrers anys ha emergit un nou model de negoci que, si ben bé no ha desplaçat els dos anteriors del tot, els ha deixat amb poca presència. Es tracta del model de micro-pagaments integrats a l'aplicació. El que abans era una versió gratuïta i limitada, ara es converteix en una versió gratuïta i limitada, a menys que l'usuari compri diferents funcionalitats directament dintre de l'aplicació. Aquest model es coneix amb el nom d'*In App Purchase*, i ha passat a ser la primera font d'ingressos a les tendes d'aplicacions (gairebé el 95% del total [2]).

Donat aquest escenari de prosperitat i innovació, i essent un mercat recent i poc explotat, nombrosos projectes han aflorat per beneficiar-se'n, creant milions d'aplicacions: aplicacions de comunicació, de gestió, d'entreteniment, jocs, etc... Aquest darrer tipus d'aplicacions, els jocs, son la categoria que més ingressos obté mitjançant aquest model de negoci, i és per això que són la categoria amb més aplicacions a la Apple Store i Google Play. Un dels projectes que ha aflorat arrel d'aquest escenari de prosperitat és el que es detalla en aquesta memòria: **Preguntas Incómodas**.

Preguntas Incómodas pretén ser un joc casual per a dispositius mòbils, basat en la dinàmica de preguntes i respostes que es pot trobar a jocs clàssics com Trivial Pursuit i les seves variants mòbils. A través de l'entreteniment dels usuaris, es pretén realitzar un projecte econòmicament viable dintre del model de negoci de micro-pagaments, i el full de ruta per aconseguir-ho es detalla al llarg d'aquesta memòria.

ESTAT DE L'ART

Resulta evident que Preguntas Incómodas no és el primer projecte que neix amb la intenció d'aprofitar l'avinentesa del mercat creixent descrit anteriorment. Per aquesta raó, ha estat fonamental realitzar un anàlisi de mercat sobre els possibles competidors que es troben al mercat, així com analitzar i trobar el factor diferencial que farà de Preguntas Incómodas el producte líder sobre la resta.

Trivia Crack

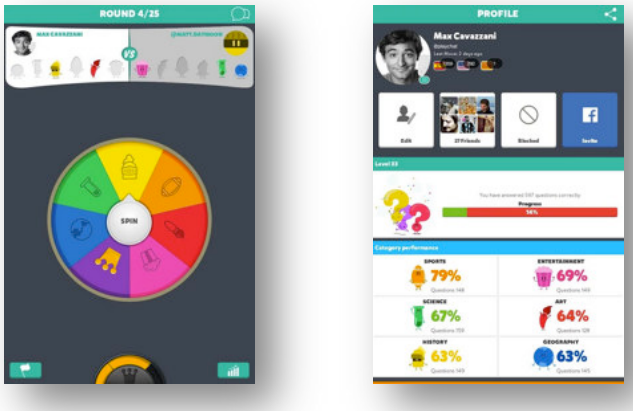


Figura 2. Captures de pantalla de Trivia Crack

Trivia Crack és una versió mòbil del clàssic Trivial Pursuit, amb preguntes classificades per temàtiques de cultura general (història, ciència, cinema, esports...).

L'objectiu d'aquest joc és guanyar la partida completant les preguntes de cada temàtica abans que el teu contrincant.

Atriviate

Figura 3. Captures de pantalla d'Atriviate

Atriviate és una altra versió del clàssic Trivial Pursuit. Les regles del joc són exactament iguals a Trivia Crack, amb la qual cosa ja s'ha esmentat el seu funcionament.

Millonario Quiz

Figura 4. Captures de pantalla de Millonario Quiz

Millonario Quiz és un joc de preguntes de cultura general amb quatre possibles respostes. L'usuari pot fer servir fins a quatre comodins per tal d'encertar totes les preguntes proposades, i la partida acaba quan l'usuari respon errròniament una d'elles. L'objectiu d'aquest joc és encertar totes les preguntes i endur-se el premi final. En aquest cas no es tracta d'un joc social, sinó d'un joc individual per posar a prova la cultura general de l'usuari.

Logo Quiz

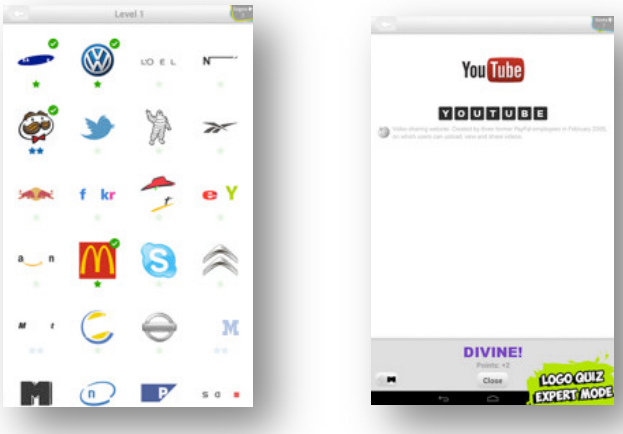


Figura 5. Captures de pantalla de Logo Quiz

Logo Quiz és un joc on l'usuari ha d'encertar la marca de tots els logotips mostrats a la pantalla. És una forma diferent de representar una dinàmica de joc de pregunta/resposta. En aquest cas, l'objectiu és encertar totes les marques per obtenir la major puntuació possible.

Havent analitzat quatre productes d'èxit candidats a ser competidors de Preguntas Incómodas, es pot concloure que la principal diferència amb el projecte proposat és la tipologia de les preguntes i l'objectiu final. En Pregunta Incómodas, l'objectiu de l'usuari és **descobrir els secrets més íntims del rival** mitjançant les preguntes de caire personal i íntim del joc, i ser la persona que més encerta les respostes del contrincant.

MOTIVACIONS

Després d'analitzar el mercat i les aplicacions similars al projecte proposat, la conclusió que s'obté és que cap aplicació ofereix una interacció social i l'objectiu de conèixer els secrets dels teus companys.

És aquí, llavors, on resideix el potencial d'aquest projecte. El motiu pel qual els usuaris jugaran a aquest joc és la **curiositat per descobrir els secrets dels seus amics**.

FUNCIONAMENT DEL JOC

Les normes bàsiques del joc, així com les diferents accions que els usuaris poden dur a terme per gaudir de tota l'experiència de Preguntas Incómodas s'exposen a continuació.

Aquest joc és un joc senzill en el qual un jugador ha de desafiar un amic (que també estigui registrat al sistema) a respondre una pregunta aleatòria de caire personal i íntim. Cal remarcar que els companys de partida han de ser coneguts (en aquest cas, amics a la xarxa social Facebook), requisit sense el qual les respostes de caire personal no disposarien d'interès social.

MODE DE JOC

Quan un jugador comença una partida amb un company, s'engega el flux de joc que consta de tres fases:

1. Tots dos jugadors han de respondre la pregunta que es mostra a l'aplicació.

2. Un cop tots dos han respost, han de tractar d'endevinar què ha respost el company.

3. Acabada la fase dos, es mostra el resultat de la partida amb la pregunta en qüestió, les respostes reals de cada jugador i si s'ha endevinat la resposta del rival.

Si un jugador ha encertat la resposta del rival, el seu percentatge de coneixement sobre l'altra persona s'incrementarà. En canvi, si falla, disminuirà.

És en aquest punt on es mostra el valor addicional de Preguntas Incómodas envers altres jocs del mateix caire:

1. Els jugadors coneixen la resposta a la pregunta, de caire personal i secret, que el seu rival ha respost.

2. Els jugadors juguen per ser la persona que més coneix un altre jugador, competint amb altres per aconseguir el major percentatge d'encerts.

FUNCIONALITATS ADDICIONALS

Com a funcionalitats addicionals que s'ofereixen a l'usuari existeix la possibilitat de publicar la pregunta i resposta d'un company de partida a les xarxes socials. Aquesta funcionalitat afegeix un factor d'intriga als usuaris al no saber si el company serà capaç de publicar la seva resposta.

A més a més, si la persona concreta amb la qual un usuari vol jugar no disposa del joc, podrà enviar-li una invitació electrònica mitjançant Facebook per descarregar-lo i jugar plegats.

Per acabar, una de les funcionalitats addicionals és la recepció de notificacions emergents, enviades des del servidor als dispositius mòbils, amb la qual cosa es pretén augmentar la retenció dels jugadors i avisar-los quan la seva interacció es requerida en una partida. Aquesta funcionalitat és totalment necessària ja que Preguntas Incómodas és un joc asíncron, és a dir, els jugadors poden respondre/endevinar preguntes en qualsevol moment i hora del dia.

OBJECTIUS

L'objectiu d'aquest projecte és oferir a l'usuari final un entreteniment que alimenta la seva curiositat envers els secrets dels seus amics, podent accedir-hi tant a través d'un dispositiu mòbil Android com iOS.

Alhora i, en l'àmbit tècnic, aquest projecte presenta diversos desafiaments en els següents aspectes:

1. Desenvolupar un sistema tecnològic asíncron i multiplataforma.

2. Desenvolupar un sistema robust i consistent amb totes les plataformes implicades.

3. Generar un programari servidor suficientment intel·ligent i automàtic per permetre l'execució de múltiples partides, incloent notificacions automàtiques.

4. Garantir l'accés a tots aquells usuaris complidors dels requisits necessaris per ser jugador del joc.

5. Distribuir el producte i adquirir usuaris a gran escala.

6. Analitzar l'ús del producte i actuar en conseqüència en la fase de producció.

ORGANITZACIÓ DE LA MEMÒRIA

Un cop exposats els objectius i l'estat de l'art del projecte, en aquesta secció es descriu l'estructura de la memòria per garantir una senzilla lectura de la mateixa.

La primera secció que ens trobem és la **Fase d'Anàlisi**, on s'analitza la viabilitat (tècnica i econòmica) del projecte, les diferents tecnologies, eines a fer servir i programari. Seguidament s'exposen els fonaments teòrics de cada una de les tecnologies utilitzades en el projecte i la organització dels components de programari.

A la **Fase de Desenvolupament,** es detalla tot el disseny de programari requerit, així com les implementacions en cadascuna de les plataformes, exemples de codi i diagrames de flux, oferint una visió més concreta del treball realitzat i la seva estructura.

Un cop detallat el desenvolupament del projecte, trobem la **Fase de Producció,** on es detalla el procés seguit per publicar el projecte i com fer-lo arribar als usuaris finals mitjançant campanyes de comunicació, inversió en publicitat interactiva, etc...

Per últim, la darrera secció contempla la **Fase de Postproducció**, que exposa les conclusions extretes de la publicació del producte, dades recol·lectades de l'ús dels usuaris, anàlisi de punts febles i treball futur.

FASE D'ANÀLISI

ANÀLISI I REQUISITS DE DISPOSITIUS

Tractant-se d'un joc per a dispositius mòbils, es fa necessari disposar d'un o diversos dispositius per tal de desenvolupar el programari, garantint el funcionament en el màxim nombre d'ells possible, o bé fent servir emuladors per a la fase de proves i qualitat.

Pel que fa a les versions dels sistemes operatius dels dispositius, el joc podrà executar-se sobre versions d'iOS superiors a la 7.0 (inclosa), i a partir de la versió 2.3 d'Android, la qual cosa garanteix el funcionament en més del 90% dels dispositius mòbils que allotgen aquests sistemes operatius.

A més a més, es requereix un ordinador configurat per desenvolupar el programari, amb sistema operatiu Mac OS (per desenvolupar la versió iOS del projecte, ja que no es permet el desenvolupament sota un altre sistema operatiu) i les eines que s'exposen a la següent secció.

ANÀLISI I REQUISITS DE PROGRAMARI

Preguntas Incómodas és un joc multi jugador, amb la qual cosa ha estat necessari dissenyar una infraestructura de programari client-servidor, és a dir, dues aplicacions que es comuniquen entre elles: una instal·lada als dispositius mòbils (client), i l'altra

allotjada a un servidor d'Internet (servidor). El client serà l'aplicació que els usuaris es podran descarregar de les diferents tendes d'aplicacions (iOS i Android), mentre que el servidor serà una única instància d'un programari que s'executarà a una màquina remota, la qual mantindrà una base de dades amb tota la informació dels jugadors, així com de les partides en curs. Aquest programari remot es comunicarà amb el client mitjançant peticions WebService a través d'Internet.

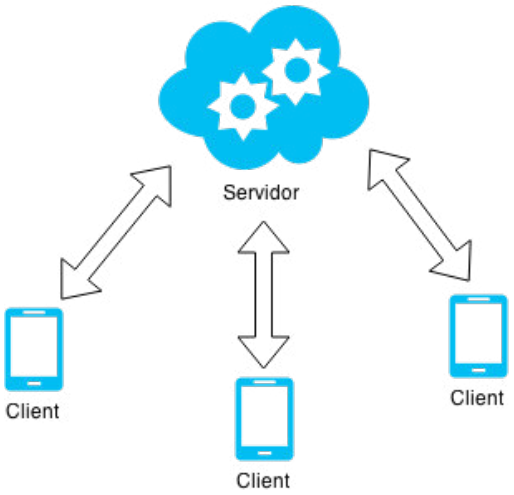


Figura 6. Diagrama de components

Com s'observa a la figura, cada aplicació client es comunica directament amb el servidor, que oferirà la informació necessària per permetre l'execució de la partida multi jugador. Amb aquesta infraestructura, el que permetem és que cada jugador pugui jugar amb un altre sense necessitat d'estar ambdós a la mateixa màquina ni de compartir una pantalla.

Un cop definida la infraestructura necessària per desenvolupar el projecte, cal realitzar un estudi de viabilitat tecnològica per tal de seleccionar les plataformes més adients a cadascun dels components de programari esmentats.

Per a la part servidora s'ha seleccionat l'eina Parse [3], una eina online que permet la fàcil i ràpida creació d'una base de dades allotjada al núvol, així com WebServices per la consulta, creació i modificació d'objectes dins la mateixa base de dades. Amb aquesta abstracció, podem centrar-nos en el disseny de la base de dades relacional i de les funcionalitats que haurà de cobrir el programari allotjat al servidor, sense necessitat de preocupar-nos per la infraestructura d'emmagatzematge i accés a les dades.

Parse ens ofereix diferents llibreries per fer ús de la seva eina a través de múltiples sistemes operatius (Android, iOS, Windows, OS X, etc...) de forma senzilla i ràpida, amb una documentació extensa i millorada al llarg dels anys. Com anècdota, remarcar que l'empresa Parse va passar a formar part del conjunt d'empreses de Facebook en Abril de 2013, la qual cosa garanteix la qualitat, potència i utilitat de l'eina.

Havent seleccionat la tecnologia i plataforma per a la part servidora, la implementació de la qual veurem més endavant, és el torn de realitzar l'estudi de viabilitat per la part client. Múltiples plataformes de creació de videojocs han estat analitzades i classificades, permetent una base sòlida per a la selecció de la plataforma a fer servir. A continuació es troba la taula derivada d'aquest anàlisi i algunes de les funcionalitats tingudes en compte.

Plataforma	Llenguatge de programació	Codi obert	Plataformes suportades	Preu	Comunitat i documentació
Cocos2d-x	C++ Javascript	Sí	iOS Android Windows Web OS X Windows Phone	Gratuït	Sí
Unity	C#	No	iOS Android Windows Web OS X Windows Phone	Gratuït [1]	Sí
Unreal Engine	C++	No	iOS Android Windows Web OS X Windows Phone	Gratuït [2]	Sí

Taula 1. Resultat d'anàlisi de plataformes de videojocs

Dues diferències importants entre **Cocos2d-x** i les altres dues plataformes analitzades són la **llicència de codi obert**, cosa que garanteix un continu suport per part de la comunitat i la possibilitat d'adaptar la tecnologia a les pròpies necessitats, i el **nul cost del seu ús**, sense necessitat de pagar per iniciar el desenvolupament ni abonar comissions lligades a la facturació del projecte.

Essent així, per desenvolupar **Preguntas Incómodas** s'ha decidit fet servir la plataforma Cocos2d-x.

Un cop decidides les tecnologies que compondran el projecte, analitzem els llenguatges de programació necessaris per desenvolupar el projecte, així com les eines disponible per dur a terme aquest desenvolupament.

[1] Gratuït per iOS i Android, amb publicitat de la plataforma. Pagament per funcionalitats extra.
[2] Gratuït fins facturar 3.000 USD per trimestre. Després, comissió del 5%.

Pel que fa al programari a desenvolupar en la part servidora, el llenguatge de programació a utilitzar en Parse és JavaScript, amb la qual cosa només cal un editor de text per crear el codi. El procés de creació i publicació es detallarà més endavant.

D'altra banda, havent conclòs que la tecnologia per la part client serà Cocos2dx, són necessaris els entorns de desenvolupament tant d'iOS com d'Android, és a dir, **XCode IDE 6** [4] i **Eclipse IDE for Java Developers** [5]. A més a més, en el cas d'Android, ens cal l'SDK que ens ofereix Google per desenvolupar aplicacions per al seu sistema operatiu, el *plugin* per connectar l'SDK amb l'entorn de desenvolupament Eclipse (**Android Developer Tools** [6]) i **l'NDK** [7], llibreria que permet connectar programari desenvolupat en Java amb mòduls desenvolupats en C++. Aquesta última llibreria es la que ens permetrà integrar la plataforma Cocos2dx al programari d'Android.

Els llenguatges de programació necessaris són **Objective C** [8] (llenguatge nadiu d'iOS), **Java** [9] (llenguatge nadiu d'Android), **C++** [10] (llenguatge per desenvolupar en Cocos2dx) i **XML** per a la interfície d'usuari i altres fitxers de configuració.

Per tant, seguint amb el diagrama de components de programari del projecte, podem detallar la següent figura:

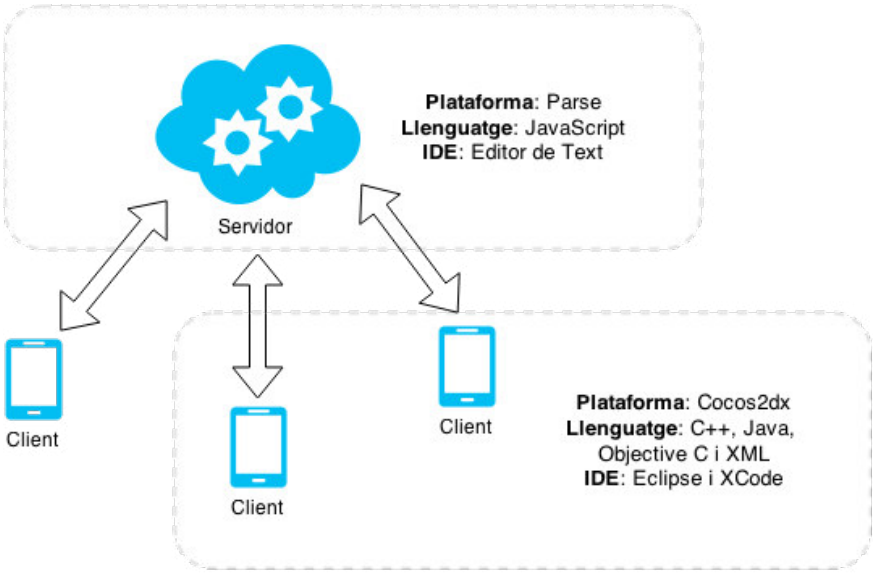


Figura 7. Diagrama de components, plataformes i llenguatges de programació

Cal mencionar que l'ús del programari de desenvolupament i les llibreries esmentades és completament gratuït, si més no l'única eina susceptible de ser de pagament és Parse, però amb la llicència gratuïta és més que suficient per al projecte que ens ateny, ja que ens permet fer 30 peticions per segon des dels clients al servidor, una capacitat d'emmagatzematge de 20GB i una capacitat de transferència de 2TB.

No serà gratuïta la publicació de les versions del programari client a les tendes d'aplicacions, tant d'Android com d'iOS, però això es detallarà més endavant a la Fase de Postproducció.

PLANIFICACIÓ

Havent finalitzat la fase d'anàlisi de les eines necessàries per a desenvolupar el projecte, en aquesta secció es realitza una proposta de planificació en termes temporals i econòmics.

A l'Annex I es troba el diagrama de Gantt on es detalla l'estimació temporal i de costos del projecte, resultant en **54 dies productius** entre els diversos perfils requerits: director de projecte, enginyer, dissenyador gràfic i executor de proves.

Pel que fa als costos del projecte per perfils, seguidament es desglossen les tasques per perfil, així com el cost de cada tasca.

DIRECTOR DE PROJECTE

Análisis y requisitos	6d	2400 €
Toma de requisitos	1d	400 €
Documentacion	5d	2000 €

ENGINYER

Implementación	48d	14400 €
Motor de juego en cliente	10d	3200 €
Motor de juego en la nube	10d	3200 €
Base de datos en la nube	4d	1280 €
Integración servicios REST	4d	1280 €
Integración Facebook	2d	640 €
Integración analíticas	1d	320 €
Pantalla de login	2d	640 €
Pantalla de menú inicial	2d	640 €
Pantalla de perfil	2d	640 €
Pantalla de juego	4d	1280 €
Pantalla de juego finalizado	2d	640 €
Pantalla de búsqueda de amigos	2d	640 €

DISSENYADOR GRÀFIC

⊟**Diseño gráfico**	20d	4800 €
Icono y marca del juego	3d	720 €
Pantalla splash	1d	240 €
Pantalla de login	2d	480 €
Pantalla de menú inicial	2d	480 €
Pantalla de perfil	2d	480 €
Pantalla de juego	4d	960 €
Pantalla de juego finalizado	4d	960 €
Pantalla de búsqueda de amigos	2d	480 €

EXECUTOR DE PROVES

⊟**Testing**	18d	2400 €
Testing versión Alpha	2d	240 €
Corrección errores Alpha	3d	960 €
Testing versión Beta	2d	240 €
Corrección errores Beta	3d	960 €
⊟**HITOS**	33d	0 €
Diseño finalizado	0d	0 €
Version Alpha	0d	0 €
Version Beta	0d	0 €
RELEASE CANDIDATE	0d	0 €

Amb aquesta planificació de tasques i estimant el cost per hora dels perfils: director de projecte, 50 EUR/hora; enginyer, 40 EUR/hora; dissenyador gràfic, 30 EUR/hora; i executor de proves 15 EUR/hora, el cost total del projecte ascendeix a 24.000 EUR. A més a més, cal sumar la quantia d'una estació de treball on desenvolupar el programari (ja que la resta d'eines de programari analitzades tenen cost nul), la qual cosa incrementa el cost total del projecte a **24.600 EUR**.

FONAMENTS TEÒRICS

En aquesta secció es detallen totes les plataformes seleccionades per a dur a terme el projecte, així com les seves principals funcionalitats i fonaments teòrics. Començant per la part servidora, seguint amb la plataforma Android (SDK i NDK) i finalitzant amb Cocos2dx, s'esmenten totes i cadascuna de les eines de programari que permeten el desenvolupament de Preguntas Incómodas.

PARSE

En primer lloc i, com ja s'ha esmentat, Parse és una eina online que permet la fàcil i ràpida creació d'una base de dades allotjada al núvol, així com WebServices per la consulta, creació i modificació d'objectes dins la mateixa base de dades.

Figura 8. Aplicació de Preguntas Incómodas en l'eina Parse

Per fer ús de l'eina, el primer pas és crear una aplicació i la seva base de dades corresponent. Per a fer-ho, només cal afegir les entitats necessàries a la secció **Core** de la nostra aplicació prèviament creada. Parse anomena aquestes entitats **Class** (Figura 9).

Figura 9. Creació de base de dades a Parse

En aquest cas, les entitats s'anomenen *User*, *Affinity*, *Answer*, *Game*, etc...

Un cop creades les entitats, es poden afegir tants camps com ens sigui necessari, ja siguin de tipus enter, cadena de caràcters, dates, etc... Parse afegeix algunes columnes predeterminades, com la data de creació, la data de la última modificació i l'identificador (clau primària de l'entitat en la base de dades relacional, que Parse gestiona automàticament) de cada fila.

Arribats a aquest punt, la base de dades al núvol està creada, configurada i llesta per fer-se servir. És possible emplenar les dades de la base de dades de forma manual, o bé esperar que siguin les pròpies aplicacions consumidores (en el nostre cas, les aplicacions client) les que vagin emplenant les taules amb la informació corresponent (per exemple, quan un usuari es registra, es crearà un registre a la taula *User* amb la informació rebuda des del client).

Per a que les aplicacions client puguin inserir, consultar o modificar aquestes dades, Parse ens ofereix dues llibreries (Android i iOS) de programari que segueixen el paradigma orientat a objectes (ja que estan desenvolupades en Java i Objective C, respectivament). Aquestes llibreries gestionen tota la comunicació client-servidor, i ens ofereixen de forma directa i transparent tota la informació allotjada al núvol. Essent així, podem obtenir qualsevol objecte de la nostra base de dades amb molt poques línies de codi.

```
ParseQuery<ParseObject> query = ParseQuery.getQuery("GameScore");
query.getInBackground("xWMyZ4YEGZ", new GetCallback<ParseObject>() {
  public void done(ParseObject object, ParseException e) {
    if (e == null) {
      // object will be your game score
    } else {
      // something went wrong
    }
  }
});
```

Figura 10. Exemple Java d'obtenció de l'objecte GameScore allotjat al núvol

A la Figura 10 s'observa un exemple d'obtenció d'un objecte allotjat al núvol amb codi Java. Cal destacar l'ús del mètode *getInBackground* de la classe *ParseQuery*, que ens permet realitzar la comunicació client-servidor en un plànol secundari, és a dir, sense bloquejar el fil d'execució principal, que als dispositius mòbils s'encarrega de dibuixar la interfície d'usuari i respondre a les seves interaccions. Aquest fil principal d'execució mai es pot bloquejar, ja que l'usuari tindria la errònia percepció de que el dispositiu ha deixat de funcionar.

Un cop resolta la petició client-servidor, al bloc de codi executat obtindrem l'objecte desitjat allotjat a la classe *ParseObject.* Aquesta classe és el pilar base de tota la infraestructura de Parse, i la que farem servir per crear, consultar, eliminar o modificar els objectes allotjats a la nostra base de dades.

La consulta de les diferents columnes de les entitats, un cop transformades a *ParseObject*, és tan senzilla com es mostra a la Figura 11.

```
int score = gameScore.getInt("score");
String playerName = gameScore.getString("playerName");
boolean cheatMode = gameScore.getBoolean("cheatMode");
```

Figura 11. Exemple d'obtenció de valors d'una entitat

Si la base de dades creada al núvol s'ha de nodrir de la informació rebuda des del client, és tan senzill com executar les línies que es mostren a la Figura 12. Cal remarcar, en aquest cas, la crida al mètode *saveEventually,* que s'encarrega d'enviar tota la informació al servidor i de desar-la al lloc corresponent.

```
ParseObject gameScore = new ParseObject("GameScore");
gameScore.put("score", 1337);
gameScore.put("playerName", "Sean Plott");
gameScore.put("cheatMode", false);
gameScore.saveEventually();
```

Figura 12. Exemple d'inserció d'una entitat a la base de dades

De la mateixa senzilla manera que s'obté o es crea un objecte, es pot modificar o eliminar.

Una altra de les funcionalitats que ofereix Parse i que es fa servir al projecte proposat és el codi executat al núvol. Aquest codi, escrit en JavaScript, es pot executar o bé directament com un WebService que es crida des de l'aplicació client (petició HTTP), o bé detonat després d'una acció realitzada a qualsevol objecte de la base de dades (aquest detonant es coneix com a *Trigger* [11] en base de dades).

Per crear aquest codi, cal instal·lar una petita eina executable a la consola de comandes. Aquesta instal·lació és tan senzilla com executar la següent comanda:

```
curl -s https://www.parse.com/downloads/cloud_code/installer.sh | sudo /bin/bash
```

Aquesta comanda descarrega i executa un petit *script* que instal·la l'eina anomenada **parse** a la carpeta /usr/local/bin, en cas de fer-ho a un entorn Unix. Un cop instal·lada, cal crear un projecte de codi al núvol, seguint la documentació que ens ofereix Parse.

```
$ parse new MyCloudCode
Email: ninja@gmail.com
Password:
1:MyApp
Select an App: 1
$ cd MyCloudCode
```

Figura 13. Creació d'un projecte de codi al núvol

A la Figura 13 es mostra com crear aquest projecte. Serà necessari indicar a l'eina les credencials d'accés a Parse (direcció de correu electrònic i contrasenya), així com l'aplicació allotjada al núvol a la que es vol associar el projecte creat. Un cop executada aquesta instrucció, l'estructura del projecte haurà estat creada tal i com es mostra a la Figura 14.

Per tal de desenvolupar el codi que s'executarà al núvol, només cal fer servir la carpeta *cloud,* on s'emmagatzemarà tot el codi JavaScript necessari, i la carpeta *public,* que contindrà tots els recursos estàtics que l'aplicació servidora requereixi.

Un cop desenvolupat el codi, cal enviar-lo a Parse per tal que l'ofereixi a la seva infraestructura al núvol. Aquest enviament (o publicació), s'ha de realitzar mitjançant *"parse deploy"*.

```
-config/
  global.json
-cloud/
  main.js
-public/
  index.html
```

Figura 14. Estructura de carpetes

```
parse deploy
```

Arribats a aquest punt, l'aplicació creada i configurada al núvol disposa d'un conjunt de programari executable. Ara bé, l'execució d'aquest codi es pot realitzar a través de *WebServices* o *Triggers*, com s'ha esmentat amb anterioritat.

Per tal de definir un servei web, només cal crear al fitxer *main.js* del projecte una porció de codi com la que es mostra a la Figura 15.

```
Parse.Cloud.define("hello", function(request, response) {
  response.success("Hello world!");
});
```

Figura 15. Servei web en JavaScript

En l'exemple de la Figura 15 es mostra la creació d'un servei web anomenat *hello*, la funcionalitat del qual no és més que retornar, en format JSON, la cadena de caràcters "*Hello world!*". Per executar aquest servei web en Android, és tan senzill com invocar al mètode *callFunctionInBackground* de la classe *ParseCloud*, disponible a la llibreria de la plataforma.

```
ParseCloud.callFunctionInBackground("hello", new HashMap<String, Object>(), new FunctionCallback<String>() {
  void done(String result, ParseException e) {
    if (e == null) {
      // result is "Hello world!"
    }
  }
});
```

Figura 16. Exemple de crida a un servei web des d'Android

D'altra banda, si el projecte ha d'executar codi al núvol en algún d'aquests escenaris: just abans d'emmagatzemar un nou objecte a la base de dades, després d'emmagatzemar-lo, just abans d'esborrar-ne un o després d'esborrar-lo; cal invocar al mètode *Parse.Cloud.beforeSave* (si es tracta del primer escenari) i definir el comportament requerit.

Per exemple, el codi de la Figura 17 mostra la definició d'un *trigger* que s'executa just abans d'emmagatzemar una revisió d'un usuari, on es comprova la integritat de la informació rebuda. En aquest cas, la puntuació de l'usuari no pot ser menor que 1 ni major que 5. D'aquesta manera, es garanteix que la base de dades sempre respecta els criteris de l'aplicació i no resulta inconsistent.

```
Parse.Cloud.beforeSave("Review", function(request, response) {
  if (request.object.get("stars") < 1) {
    response.error("you cannot give less than one star");
  } else if (request.object.get("stars") > 5) {
    response.error("you cannot give more than five stars");
  } else {
    response.success();
  }
});
```

Figura 17. Exemple de *trigger*

Amb aquestes funcionalitats: creació i modificació d'una base de dades al núvol, creació de serveis web i execució de codi automàtic (*triggers*), Parse cobreix tots els requeriments definits al projecte Preguntas Incómodas, concretament amb els requeriments de la part servidora.

Un cop coberts els requeriments de la part servidora, en les tres següents seccions es detallen les eines necessàries (i el seu ús) per a la creació de l'aplicació client:

primerament s'esmentarà l'SDK d'Android, la plataforma que ofereix Google per a la creació d'aplicacions al seu sistema operatiu mòbil, tot seguit es parlarà del component NDK d'aquesta mateixa plataforma, que permet executar codi C/C++ als dispositius Android. Finalment, es detallarà l'ús i funcionament de Cocos2dx, plataforma sobre la qual es desenvolupa la gran part del programari client.

SDK ANDROID

L'SDK d'Android és una plataforma de programari sobre la qual es poden construir aplicacions funcionals als dispositius amb el sistema operatiu mòbil de Google: Android.

El 23 de setembre de 2008, Google publicava la versió 1.0 del seu sistema operatiu sobre l'exclusiu dispositiu HTC Dream. No obstant, no va ser fins el 15 de setembre de 2009 que es va publicar la primera versió de l'SDK, que permetria als desenvolupadors crear les seves pròpies aplicacions. Aquest SDK es publicava de la mà de la versió 1.6 d'Android (anomenada Donut), essent aquesta la quarta versió publicada per l'empresa americana.

Des de llavors, tant les venidores versions d'Android com de l'SDK han incorporat noves funcionalitats i característiques atenent a la demanda dels usuaris i l'evolució de la tecnologia mòbil. Tal és el punt que, a dia d'avui, Android es troba a la versió 5.1, anomenada Lollipop, i l'SDK l'acompanya amb un nombre de versió 22 (el que els desenvolupadors anomenen nivell d'API – *Application Programming Interface*). Arrel de totes aquestes versions, sorgeix la problemàtica de la fragmentació, és a dir, hi ha milers de dispositius al món amb diferents versions d'Android, amb la qual cosa es fa indispensable que cada nova versió del sistema operatiu sigui retro-compatible amb les aplicacions programades per a versions anteriors i, pel moment, així és.

Actualment, l'SDK d'Android permet desenvolupar aplicacions amb connexió directa a Internet, ja sigui mitjançant tecnologia 3G o WiFi, així com comunicacions *bluetooth*, sensors de geolocalització, compres integrades en aplicacions, comunicacions NFC (*Near Field Communication*), etc... Malgrat tot, per al projecte descrit només cal disposar d'una connexió permanent a la xarxa d'Internet i el mòdul anomenat NDK, que permet l'execució de codi C/C++ en una aplicació Android, programada principalment en codi Java i XML.

El procés de creació d'una aplicació bàsica per al sistema operatiu Android és molt senzill, i comença per la instal·lació dels components necessaris: Eclipse IDE, on es desenvoluparà el projecte, Android SDK, la plataforma de Google, i un mòdul anomenat ADT (Android Developer Tools), que conté les eines necessàries per a lligar el programari desenvolupat amb un dispositiu real amb el sistema operatiu Android. Aquests dos últims s'instal·len a l'estació de treball i s'informa a Eclipse de la seva localització en el sistema de fitxers, de tal manera que el procés de compilació i execució és completament transparent per al programador.

No obstant, donada la magnitud d'aquest projecte i les avançades eines de programació (com l'NDK), cal esmentar aquest procés de compilació i generació del producte final.

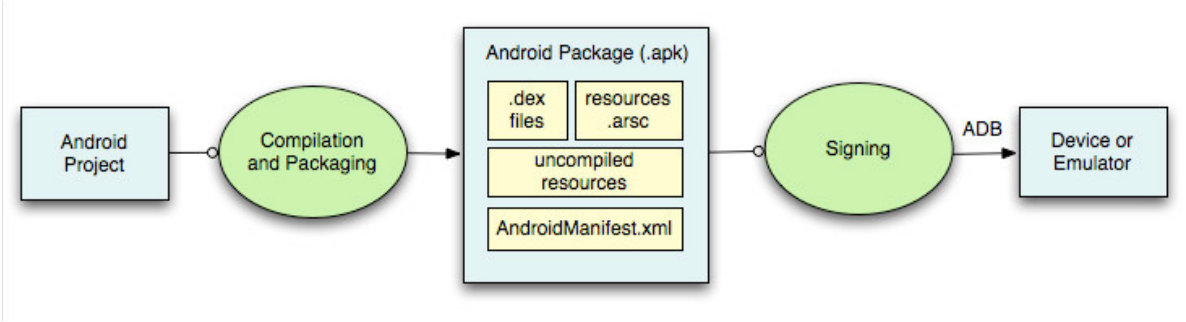


Figura 18. Procés de generació d'una aplicació Android

A la Figura 18 es descriu aquest procés. A partir d'un projecte desenvolupat en Eclipse, les eines disponibles a l'Android SDK compilen i encapsulen tot aquest projecte, obtenint un fitxer amb extensió **APK** (Android Package), que conté tots els recursos i el codi de l'aplicació, ja processat i comprimit. Aquest fitxer és instal·lable 100% a qualsevol dispositiu Android (amb el nombre de versió adequat), però per tal de ser distribuït a través de Google Play (tenda oficial d'aplicacions), cal realitzar un signat previ amb el qual el desenvolupador queda associat a l'aplicació, preservant el seu dret intel·lectual així com la possibilitat de publicar futures actualitzacions.

Alguns dels components que s'observen a l'interior del fitxer APK són molt interessants, i cal detallar-los. El primer d'ells mostra uns fitxers de tipus **DEX**. Aquests fitxers contenen tot el codi Java compilat amb una màquina virtual clàssica Java, però afegint-hi un pas final en el qual el *bytecode* que genera la màquina virtual a partir del codi Java, és comprimit i compilat per una màquina virtual Dalvik, que s'encarrega d'optimitzar el codi per tal de ser executat en màquines amb prestacions limitades, com pugui ser poca capacitat d'emmagatzematge, poca memòria virtual, dispositius amb poca bateria, etc...

Tot seguit, els recursos contenen totes les imatges, literals i gràfics requerits per l'aplicació per ser més agradable a l'usuari final i facilitar el seu ús. Per últim, el fitxer **AndroidManifest.xml**, de format XML, que conté tota la informació necessària per a identificar l'aplicació (nom, identificador únic, nombre de versió, recursos del dispositiu que requereix, permisos que l'usuari final haurà d'acceptar, etc...).

Al fitxer AndroidManifest.xml també s'exposen tots els artefactes propis del sistema operatiu que formen part d'una aplicació Android: Activity, BroadcastReceiver, Permissions, Service, etc... De tots aquests artefactes, a Preguntas Incómodas se'n fan servir els tres primers. Les Activities són el pilar principal de l'aplicació, i especifiquen el flux d'execució de la mateixa, així com la interfície gràfica que s'ha de mostrar en cada instant. Una Activity es pot definir com una unitat aïllada de responsabilitat que s'encarrega de mostrar una interfície gràfica i oferir a l'usuari final unes funcionalitats concretes, com bé pugui ser accedir a l'aplicació, mostrar un llistat de contactes o trucar a un telèfon determinat.

Els BroadcastReceivers són components dedicats a escoltar les notificacions que el sistema operatiu llença a totes les aplicacions mentre el dispositiu està engegat. Aquests tipus de notificacions van des de informar que el dispositiu s'acaba d'engegar, fins a notificacions de bateria baixa, trucada entrant (si escau), auriculars

connectats, etc... En el cas de Preguntas Incómodas, aquest component es fa servir per detectar la instal·lació de l'aplicació en un dispositiu nou i l'arribada de notificacions push (s'esmenten a l'especificació de la part servidora d'aquesta memòria).

Per últim, els permisos són accessos a recursos que l'usuari final ha de consentir per tal de fer servir l'aplicació, com per exemple l'accés a Internet, l'accés a la posició GPS del dispositiu, la vibració, etc... Aquests permisos es mostren abans de que l'aplicació sigui instal·lada, permetent a l'usuari rebutjar-los i evitar accessos no desitjats a la seva informació.

Aquests permisos defineixen la política de seguretat del sistema operatiu i les seves aplicacions. Basant-se en les lleis de seguretat dels sistemes UNIX (un procés només pot accedir a la seva porció de memòria, o bé als recursos públics), l'usuari té tot el poder de decisió sobre el programari que fa servir o que instal·la, exposant els recursos i les dades del seu dispositiu en base als permisos definits a l'aplicació. Per exemple, una aplicació pot demanar permís per accedir a la galeria fotogràfica. L'usuari haurà de garantir aquest permís per tal que l'aplicació funcioni, o bé refusar-lo i no instal·lar el programari.

Tornant a les Activities, el desenvolupador en selecciona una per tal que sigui aquesta el punt d'entrada principal de l'aplicació.

```
<activity
    android:name="org.cocos2dx.cpp.AppActivity"
    android:configChanges="orientation"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >

    <!-- Tell NativeActivity the name of our .so -->
    <meta-data
        android:name="android.app.lib_name"
        android:value="cocos2dcpp" />

    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Figura 19. Activity principal de Preguntas Incómodas

A la figura superior (19) es mostra un fragment del fitxer AndroidManifest.xml de Preguntas Incómodas. Aquest fragment informa de l'Activity principal, la qual és el punt d'entrada d'execució de l'aplicació client (org.cocos2dx.cpp.AppActivity). Cal mencionar el bloc "intent-filter", que és el que defineix aquesta Activity com a principal. La resta de components formen part del desenvolupament del projecte i es detallaran en la secció corresponent. No obstant, sí cal mencionar el bloc "meta-data", on s'especifica el nom de la llibreria C/C++, desenvolupada amb Cocos2dx, i compilada per ser carregada al projecte mitjançant l'NDK.

NDK ANDROID

L'NDK d'Android (*Android Native Development Kit*) es defineix com un conjunt d'eines que permeten l'acoblament de components de programari desenvolupats en C/C++ a les aplicacions Android. L'ús d'aquestes eines està recomanat per a programari d'alt consum de CPU, com simuladors de física, processadors de senyal o motors de joc, com el projecte exposat en aquesta memòria. Llavors, l'NDK es presenta com un component necessari per al desenvolupament de Preguntas Incómodas.

La comunicació entre les dues vessants de l'aplicació: el codi nadiu (C++) i el codi propi d'Android (Java), es realitza a través d'una interfície definida, anomenada JNI (Java Native Interface). Aquesta interfície no és pròpia d'Android, sinó que es fa servir per permetre l'execució de codi compilat dintre una màquina virtual de Java.

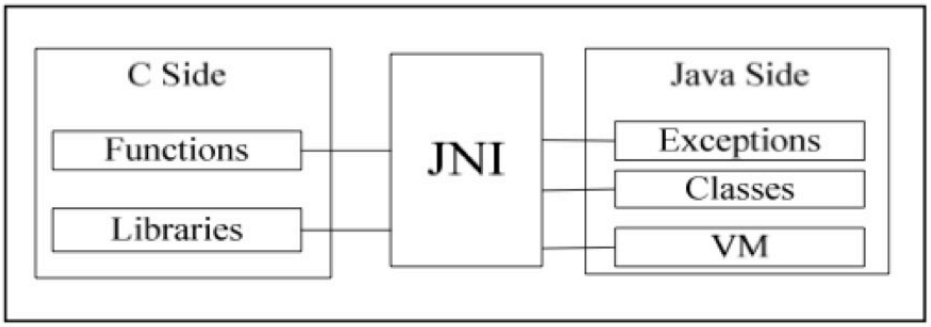


Figura 20. Diagrama de comunicació C++ - Java mitjançant JNI

Per tal de fer servir aquesta eina, cal introduir a l'estructura del projecte Android una carpeta anomenada *jni*, on es troben els fitxers *makefile (Android.mk* i *Application.mk)* i les classes d'entrada/sortida de missatges entre els dos llenguatges de programació.

Per tal de crear aquestes classes d'entrada/sortida amb JNI, primerament cal escriure una classe en codi Java, que serà la que es comunicarà amb C++ (Figura 21).

```
public class HelloJNI {
   static {
      System.loadLibrary("hello"); // Load native library at runtime
                                   // hello.dll (Windows) or libhello.so (Unixes)
   }

   // Declare a native method sayHello() that receives nothing and returns void
   private native void sayHello();

   // Test Driver
   public static void main(String[] args) {
      new HelloJNI().sayHello();  // invoke the native method
   }
}
```

Figura 21. Classe Java exemple

A la classe HelloJNI d'exemple, es pot observar una definició estàtica amb la sentència **System.loadLibrary("hello");**. Aquí s'especifica el nom de la llibreria compilada en C++ i que es carregarà a la màquina virtual.

Cal remarcar també la definició d'un mètode **native**. Aquesta especificació informa que el mètode serà desenvolupat en codi nadiu. Per últim, el mètode **main** de la classe d'exemple fa una crida a aquest mètode nadiu.

Un cop creada la classe **HelloJNI.java**, cal compilar-la mitjançant l'eina de Java javac:

```
javac HelloJIN.java
```

Aquesta instrucció generarà un *bytecode* Java al fitxer **HelloJNI.class**. I és en aquest punt quan, a través de l'eina **javah** i a partir del *bytecode*, es pot generar un fitxer de capçalera de codi C/C++:

```
javah HelloJNI
```

Aquesta última instrucció genera un fitxer anomenat **HelloJNI.h**, ja en codi C/C++ (Figura 22).

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class HelloJNI */

#ifndef _Included_HelloJNI
#define _Included_HelloJNI
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     HelloJNI
 * Method:    sayHello
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_HelloJNI_sayHello(JNIEnv *, jobject);

#ifdef __cplusplus
}
#endif
#endif
```

Figura 22. Fitxer de capçalera HelloJNI.h

Com es pot observar al fitxer de capçalera, s'ha definit el mètode **Java_HelloJNI_sayHello**, que correspon al mètode marcat com a **native** a la classe Java. Els dos paràmetres (*JNIEnv* i *jobject*) són una referència al context de JNI per tal de fer servir les funcions que ens facilita, i una referència a l'objecte **this** de la classe Java, respectivament.

Arribats a aquest punt, només cal crear el fitxer de codi font **HelloJNI.c** i definir la implementació del mètode *sayHello.*

```
#include <jni.h>
#include <stdio.h>
#include "HelloJNI.h"

// Implementation of native method sayHello() of HelloJNI class
JNIEXPORT void JNICALL Java_HelloJNI_sayHello(JNIEnv *env, jobject thisObj) {
   printf("Hello World!\n");
   return;
}
```

Figura 23. Fitxer de codi HelloJNI.c

A la Figura 23 es mostra aquest fitxer i la implementació del mètode nadiu. Cal remarcar que en aquest punt es pot fer servir qualsevol classe, llibreria o components programats en C/C++. En el cas de l'exemple únicament s'imprimeix per pantalla un missatge.

Per acabar, és necessari informar d'aquest nou fitxer de codi al fitxer *Android.mk* per tal que l'NDK el tingui en compte a l'hora de generar l'aplicació Android final.

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE := hello_shared

LOCAL_MODULE_FILENAME := libhello

LOCAL_SRC_FILES := HelloJNI.c

include $(BUILD_SHARED_LIBRARY)
```

Figura 24. Fitxer Android.mk

Per cada classe, codi o fitxer nou de C/C++ que es generi, caldrà informar al fitxer *Android.mk* després de processar la classe Java a través de la interfície JNI. En aquest exemple, al codi Java, només caldria fer una crida a **new HelloJNI().sayHello()** per executar el codi C/C++.

Gràcies a aquesta interfície d'interconnexió entre els dos llenguatges de programació, Cocos2dx és capaç d'incloure tot un sistema de classes i components propis d'un motor de videojocs a les aplicacions Android.

COCOS2DX

Cocos2d-x és una plataforma de desenvolupament de programari, creada en C++ i de codi obert, que permet als desenvolupadors de videojocs crear aplicacions per a múltiples plataformes: Android, iOS, OS X, Web, Windows i Windows Phone, i tot això sota un mateix codi, sense necessitat d'utilitzar les tecnologies de desenvolupament pròpies de cada plataforma.

Creada a mitjans de 2010, la plataforma ha anat creixent i guanyant seguidors, superant els 400.000 desenvolupadors en 2014 [12] i convertint-se en una de les principals plataformes de creació de videojocs per a dispositius mòbils.

Gràcies a la ja esmentada interfície JNI i l'Android NDK, Cocos2dx es pot executar tan en dispositius Android com iOS, plataformes objectiu del projecte exposat en aquesta memòria.

A l'hora de crear un projecte, Cocos2dx ens ofereix una eina que crearà l'estructura bàsica del projecte (Figura 25).

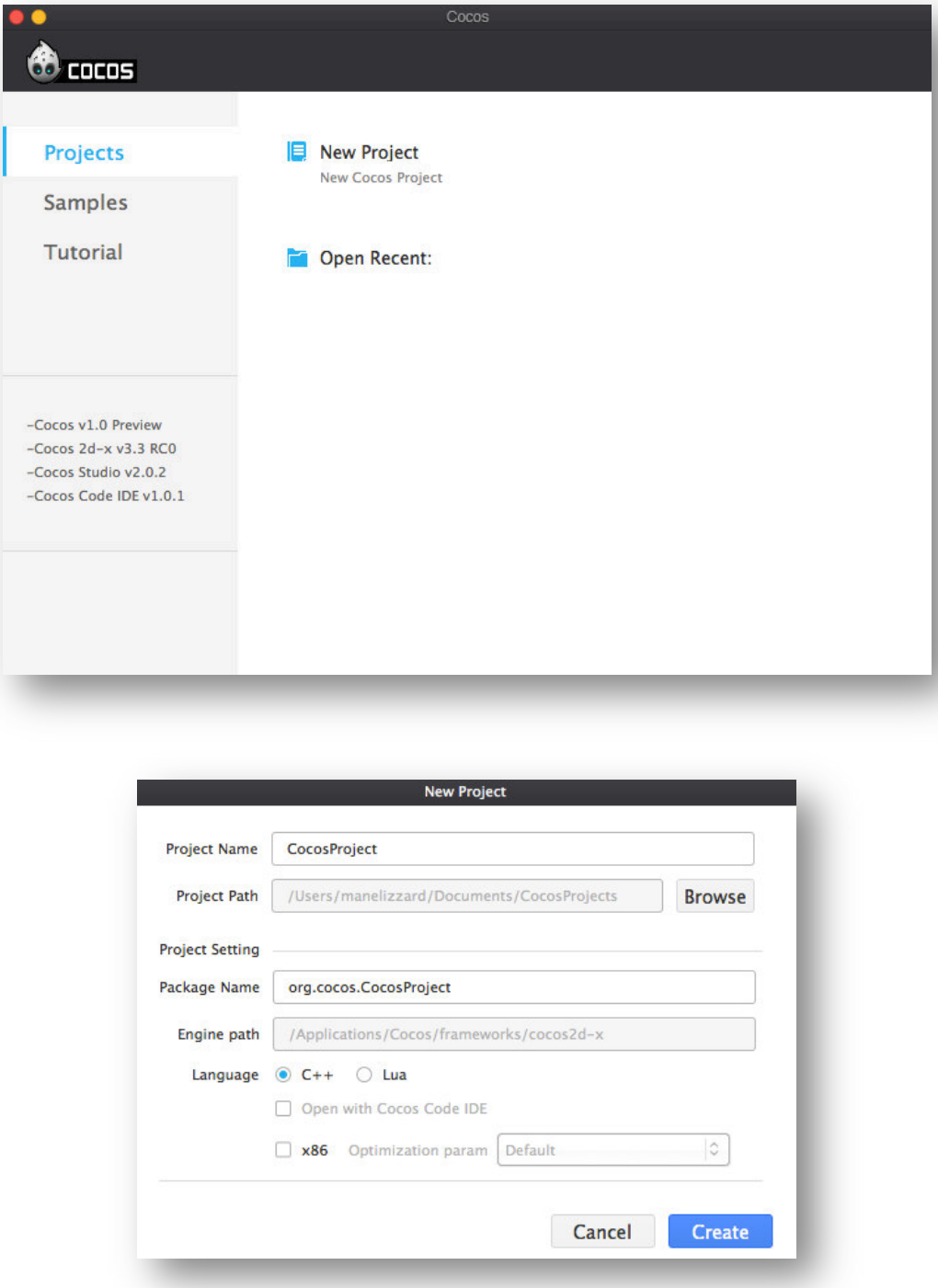


Figura 25. Eina de creació de projectes Cocos2dx

L'estructura del projecte Cocos2dx creat amb l'eina esmentada es mostra a la Figura 26.

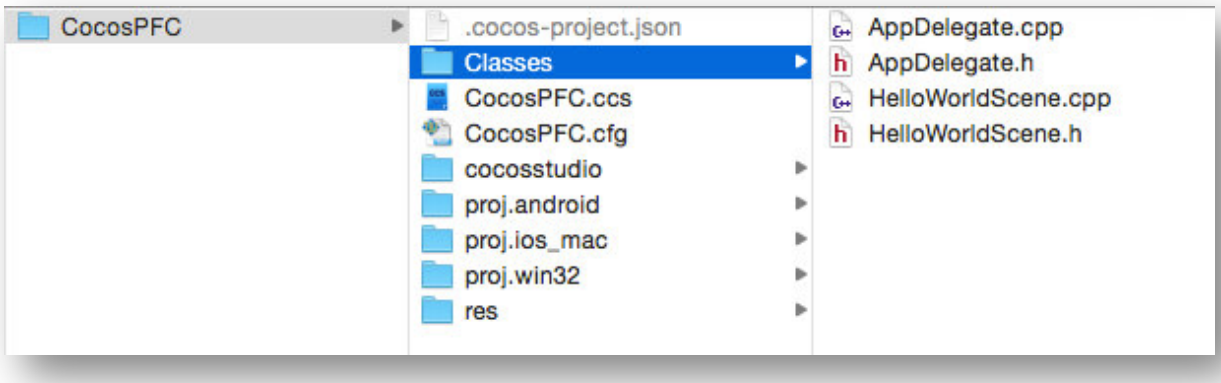


Figura 26. Estructura d'un projecte Cocos2dx

La carpeta contenidora de tot el codi C/C++ que formarà el videojoc final és la carpeta **Classes**. Dintre d'aquesta carpeta es pot crear qualsevol jerarquia de subcarpetes per tal d'organitzar millor el codi.

A la figura s'observen els directoris **proj.android, proj.ios_mac** i **proj.win32.** Aquests tres directoris contenen l'estructura bàsica d'un projecte per Android, iOS i Windows, respectivament, amb la particularitat que aquests projectes ja contenen la llibreria Cocos2dx i la carreguen de forma automàtica.

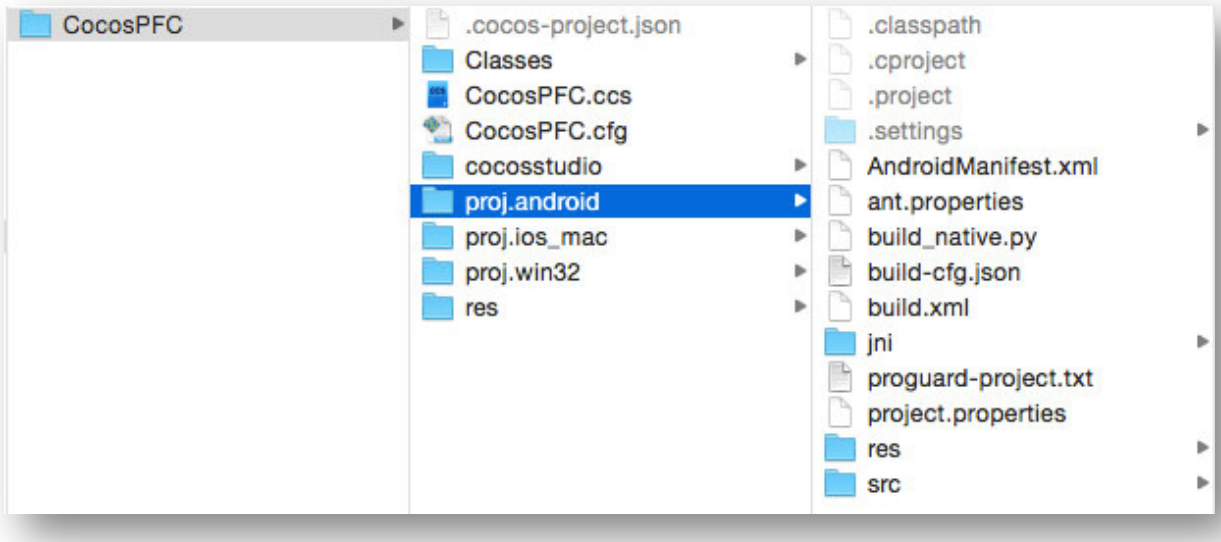


Figura 27. Projecte Android amb Cocos2x

Per exemple, al directori del projecte Android trobem el fitxer **AndroidManifest.xml**, detallat en la secció anterior, i el directori **jni**, que conté el fitxer **Android.mk** per tal de generar la llibreria C++ que, posteriorment, s'executarà des de codi nadiu d'Android.

El fitxer **CocosPFC.ccs** és un fitxer propi de l'eina Coco Studio [13], que facilita la creació d'interfícies gràfiques que s'exporten a fitxers interpretables pel motor del videojoc.

Degut a la ràpida evolució de la plataforma i a l'aparició d'eines com Coco Studio, i tenint en compte la data d'execució del projecte, les interfícies gràfiques de Preguntas Incómodas estan implementades directament sobre el codi C++, sofrint els desavantatges de l'acoblament i la difícil reutilització de components, tot i que no nul·la.

Sigui quina sigui la plataforma objectiu del projecte Cocos2dx, el punt d'entrada del codi a desenvolupar pels creadors de videojocs comença a la classe **AppDelegate,** ubicada a l'arrel del directori **Classes** del projecte. Aquesta classe s'encarrega de inicialitzar tots els components necessaris i posar a disposició del programador el flux principal del motor del joc. És responsabilitat del programador, llavors, executar des d'aquí la primera escena del joc en qüestió.

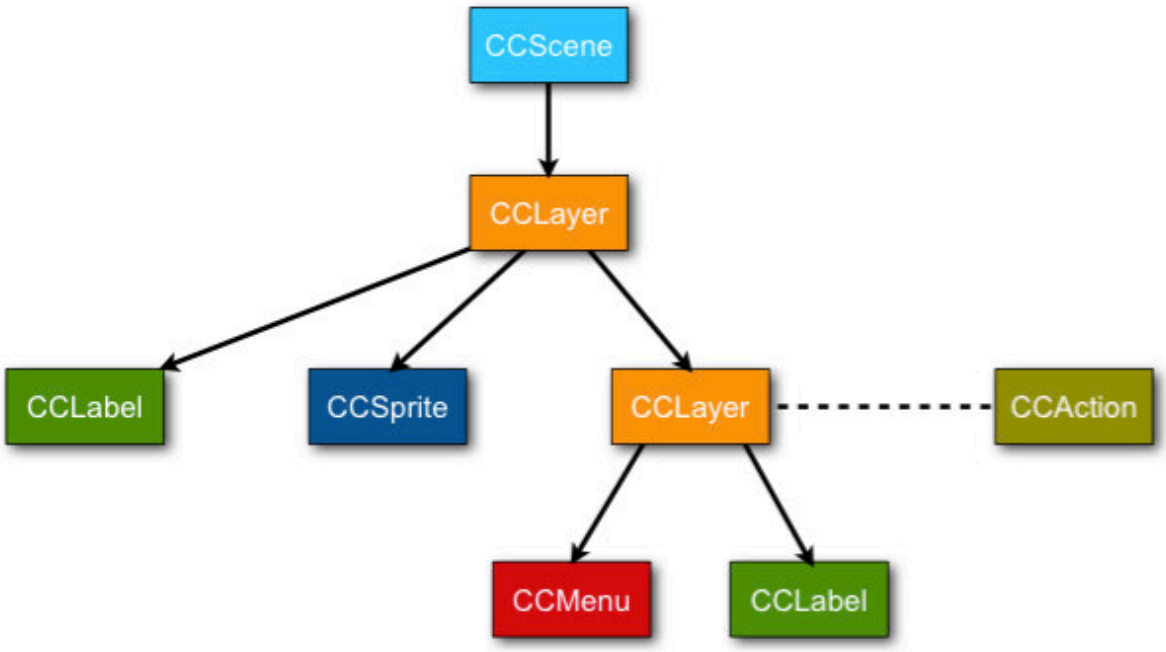


Figura 28. Graf de components d'una escena Cocos2dx [14]

L'escena executada des de la classe AppDelegate es pot definit com una unitat atòmica de responsabilitat, amb una interfície gràfica i resposta a la interacció de l'usuari.

La interfície gràfica de l'escena es crea a través de capes (CCLayer), així com *sprites* (elements gràfics de dues dimensions que s'inclouen a l'escena) i components interactuables (menús, botons, etiquetes...). Un exemple es pot trobar a la Figura 29, on s'aprecia l'escena **HelloWorld** creada de forma automàtica quan es crea un projecte Cocos2dx.

```cpp
#include "HelloWorldScene.h"
#include "cocostudio/CocoStudio.h"
#include "ui/CocosGUI.h"

USING_NS_CC;

using namespace cocostudio::timeline;

Scene* HelloWorld::createScene()
{
    // 'scene' is an autorelease object
    auto scene = Scene::create();

    // 'layer' is an autorelease object
    auto layer = HelloWorld::create();

    // add layer as a child to scene
    scene->addChild(layer);

    // return the scene
    return scene;
}

// on "init" you need to initialize your instance
bool HelloWorld::init()
{
    //////////////////////////////
    // 1. super init first
    if ( !Layer::init() )
    {
        return false;
    }

    auto rootNode = CSLoader::createNode("MainScene.csb");

    addChild(rootNode);

    auto closeItem = static_cast<ui::Button*>(rootNode->getChildByName("Button_1"));
    closeItem->addTouchEventListener(CC_CALLBACK_1(HelloWorld::menuCloseCallback, this));

    return true;
}
```

Figura 29. Escena HelloWorld

Aquesta escena s'inicialitza llegint el fitxer de interfície gràfica "MainScene.csb", creat a partir de Coco Studio. Tot seguit, s'especifica la funcionalitat del botó "Button_1", que haurà de tancar l'aplicació.

D'aquesta manera i fent servir tots els components que ofereix Cocos2dx es pot crear quasi la totalitat de videojocs 2D que hi ha actualment al mercat, com Preguntas Incómodas.

SEGURETAT

Pel que fa a la seguretat del projecte exposat en aquesta memòria, no ha estat necessari realitzar cap implementació addicional a la seguretat tècnica inherent a les eines i plataformes utilitzades: Parse, Android i C++.

Les comunicacions client-servidor realitzades a través de la llibreria de Parse es realitzen sota una estricta codificació mitjançant SSL [15], amb la qual cosa es garanteix la seguretat i confidencialitat de les dades enviades i rebudes.

En quan a la seguretat del codi font, s'empren tècniques d'ofuscació pel codi escrit en el llenguatge Java, mentre que la llibreria C++ conté la suficient seguretat al ser codi compilat, i no interpretat.

Pel que fa a la seguretat no qualificable com tècnica, és a dir, les dades personals dels usuaris emmagatzemades a la base de dades al núvol, el projecte compleix

estrictament la LOPD [16] i s'informa pertinentment als usuaris a través d'un enllaç dels termes i condicions d'ús, així com a la política de privacitat.

Aquest enllaç és accessible just abans que l'usuari pugui realitzar cap acció dintre de l'aplicació client, així com d'ésser enregistrat al sistema, assumint la lectura i acceptació per part del mateix individu (Figura 30).

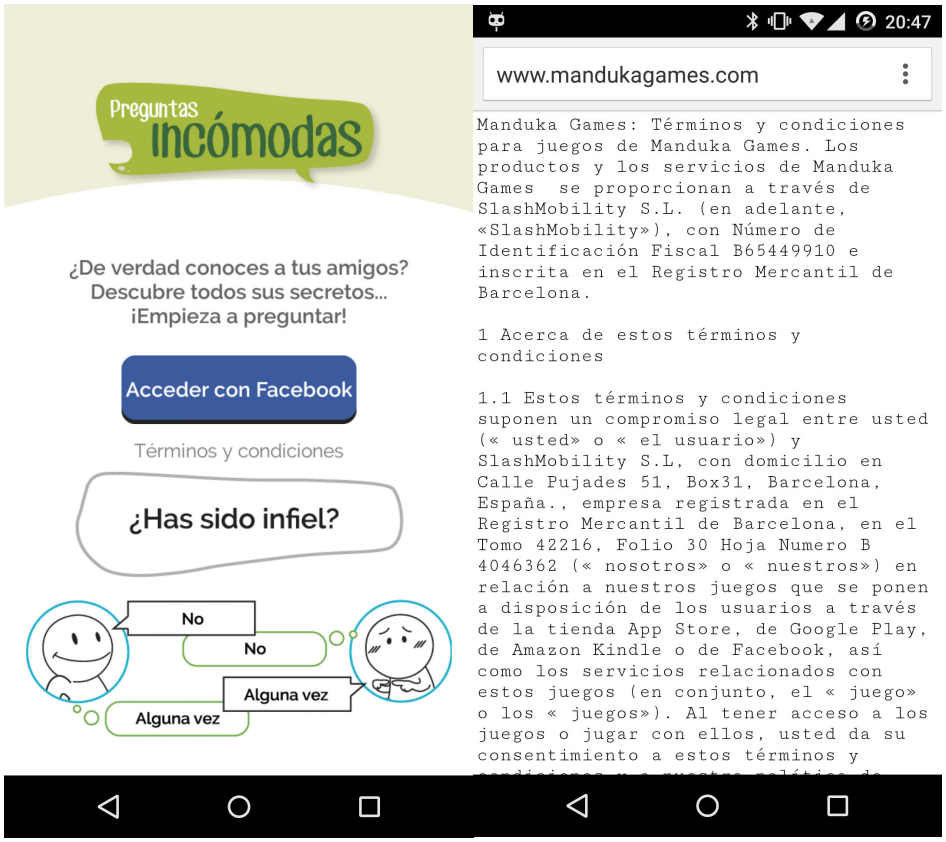


Figura 30. Termes i condicions de Preguntas Incómodas

FASE DE DESENVOLUPAMENT

Arribats a aquest punt, ja s'ha esmentat la naturalesa del projecte, els productes similars que es troben al mercat, els objectius que es persegueixen i les eines necessàries per dur-lo a terme.

Tot seguit, en la fase de desenvolupament, es detalla en profunditat com s'ha dissenyat el programari, així com la seva implementació.

La primera subsecció d'aquesta fase (Disseny de Programari) exposa el disseny de les normes del joc, és a dir, com es poden traslladar les normes a un sistema informàtic, així com el disseny realitzat per a la part servidora del projecte i la part client (Android i iOS).

Per finalitzar, la darrera subsecció tracta la implementació del disseny exposat, fent especial èmfasi en aquelles implementacions claus per al projecte, les seves estructures i diverses curiositats trobades al llarg del desenvolupament.

DISSENY DE PROGRAMARI

Per començar amb el disseny de programari cal, abans de tot, especificar com les normes del joc es veuran reflectides en el sistema informàtic, ja que tot el programari desenvolupat girarà entorn aquestes normes i lleis.

NORMES DEL JOC

Les normes del joc són senzilles i no requereixen un disseny complex, tot i que el requeriment de joc asíncron obliga a dissenyar una màquina d'estats per representar l'estat d'una partida, garantint la consistència del sistema.

MÀQUINA D'ESTATS D'UNA PARTIDA

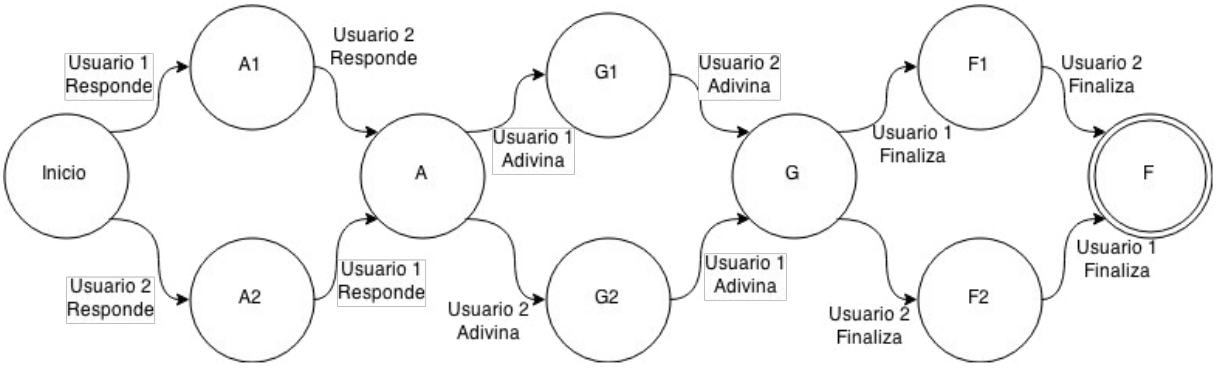


Figura 31. Màquina d'estats

A la Figura 31 es pot observar la màquina d'estats dissenyada per conèixer l'estat d'una partida en tot moment. Una partida que no es trobi en aquesta màquina d'estats es considera inconsistent i, per tant, un error d'execució.

Primerament, serà necessari que un usuari comenci una nova partida, entrant a l'estat zero (Inicio). Per últim, els dos usuaris han d'haver visualitzat el resultat de la partida abans que aquesta es consideri finalitzada. De nou, per garantir la consistència i fer més senzill el projecte, en aquest punt s'introdueix un requisit que evita que dos usuaris mantinguin dues partides diferents simultànies. Així, abans de llançar una nova pregunta al mateix company, s'ha d'haver finalitzat la pregunta prèvia.

La part servidora, un cop ha detectat un canvi d'estat d'una partida, informarà mitjançant notificacions remotes els dos clients implicats. Aquestes notificacions es llançaran en els següents estats i persegueixen els següents objectius:

1. Al començar una partida (Inicio) es notifica al company desafiat per tal que s'assabenti de la creació d'una nova partida.

2. Quan els dos usuaris han respost la pregunta (A) i es passa a la fase d'endevinament, es notifica al jugador que no ha provocat aquest salt d'estat (ja que al que l'ha provocat se'l redirigeix directament a la pantalla d'endevinament).

3. Quan els dos usuaris han tractat d'endevinar la resposta del company (G), es notifica, de nou, a l'usuari que no ha provocat el salt d'estat.

4. Quan els dos usuaris han visualitzat el resultat de la partida, notificant així el final de la mateixa i la possibilitat de crear-ne una de nova.

Amb la màquina d'estats i la definició dels instants de notificació es garanteix la consistència i correcte funcionament de tot el sistema.

La interacció dels dos clients (d'ara endavant, A i B; el que inicia la partida i l'usuari reptat) amb el servidor segueix el següent flux:

FLUX DE COMUNICACIÓ CLIENTS-SERVIDOR

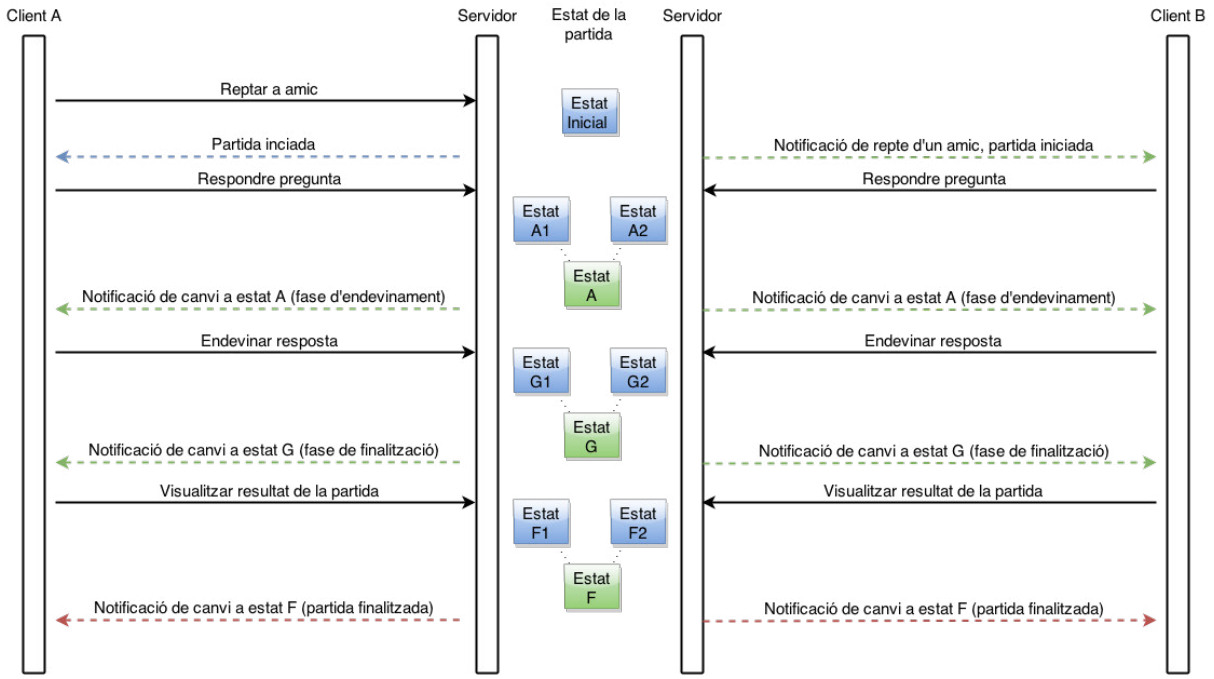


Figura 32. Flux de comunicació clients-servidor

El flux de comunicació de la Figura 32 mostra la comunicació entre els clients i el servidor, així com els estats pels quals passa la partida.

Com s'observa, tots dos clients poden realitzar la mateixa acció en indiferent ordre. Quan tots dos han realitzat l'acció, llavors es passa al següent estat i es notifica als clients (si escau).

SERVIDOR

En aquesta secció, dedicada exclusivament al disseny del programari realitzat en la part servidora, es detalla el model de dades relacional materialitzat al projecte, així com els diversos mòduls de programari que formen part del servidor. També s'exposa el flux d'execució que regeix el codi del servidor, amb les diferents operacions d'entrada i sortida d'informació.

BASE DE DADES RELACIONAL

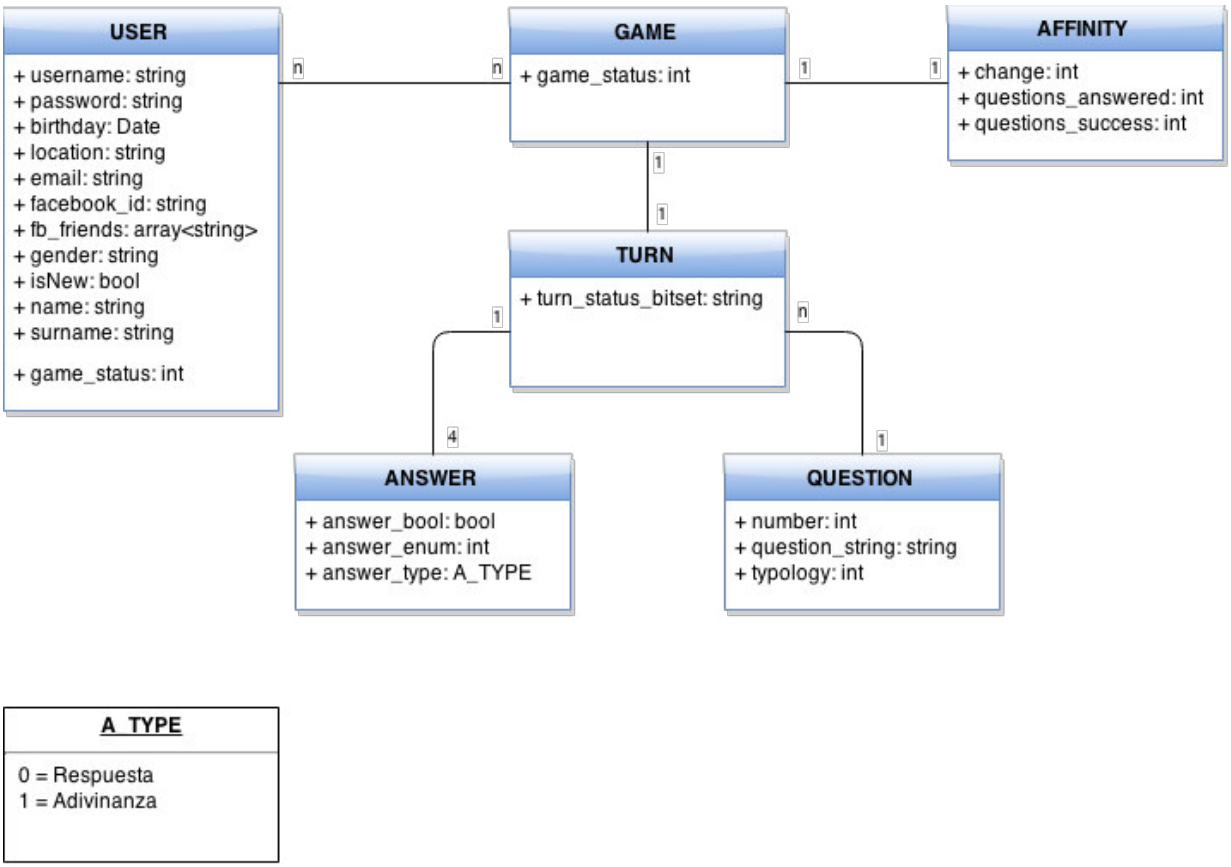


Figura 33. Base de dades relacional

El disseny de la base de dades relacional que ha de donar suport a Preguntas Incómodas es troba a la Figura 33. Els usuaris (**USER**), a més de dades importants com el nom, cognom, sexe i localització, requereixen dues dades obligatòries: **facebook_id** i **fb_friends**. El primer d'aquests dos camps emmagatzema l'identificador de l'usuari a la xarxa social Facebook, mentre que el segon emmagatzema els identificadors dels amics de l'usuari a Facebook que també són usuaris de Preguntas Incómodas.

Aquest llistat d'amics s'emplena sempre que un usuari nou es registra amb els amics ja registrats al joc. Alhora, aquests amics modifiquen el seu llistat de **fb_friends** afegint-hi el nou usuari registrat, i el servidor els hi envia una notificació per tal de informar-los que un nou amic està disponible per reptar al joc.

Un usuari podrà crear *N* jocs (**GAME**), però sempre amb usuaris diferents (aquesta restricció s'aplica a nivell de codi, ja que no es pot especificar a nivell d'*UML*).

D'aquesta entitat en neixen dues: **AFFINITY** i **TURN**. La primera d'elles contindrà la informació sobre la coneixença dels dos usuaris del joc (càlcul que es realitza a través de les respostes registrades i les encertades). Pel que fa a la segona entitat, s'ha decidit afegir-la al disseny per tal de permetre, en un futur, partides de més d'un torn (actualment, una partida disposa d'un únic torn).

L'entitat **TURN** contindrà la codificació de la màquina d'estats de la partida, representada amb un conjunt de bits de 6 elements: $B_1B_2B_3B_4B_5B_6$, on:

Bit	***Significat (0 = fals, 1 = verdader)***
B_1	L'usuari principal ha respost la pregunta
B_2	L'usuari desafiat ha respost la pregunta
B_3	L'usuari principal ha enviat la seva aposta d'endevinament
B_4	L'usuari deasfiat ha enviat la seva aposta d'endevinament
B_5	L'usuari principal ha visualitzat el final
B_6	L'usuari desafiat ha visualitzat el final

Una de les dues entitats restants, **ANSWER,** pot ser de dos tipus (resposta o endevinança), i el nombre màxim que un torn pot contenir són 4, tal i com detallen les normes del joc (dues respostes i dues endevinances). L'altra, **QUESTION,** és una entitat completament independent que emmagatzema totes les preguntes del joc i que es selecciona de forma aleatòria quan un usuari crea una nova partida. Cal remarcar en aquesta última un camp dedicat a la tipologia de la pregunta, ja que en un futur es planteja donar suport a múltiples tipologies de pregunta (preguntes de sí/no, preguntes amb quatre possibles respostes, preguntes amb resposta lliure, etc...).

MÒDULS DE PROGRAMARI AL SERVIDOR

El programari desenvolupat al servidor segueix un estricte disseny de mòduls que permeti una adequada organització del codi i funcionalitats. Essent Parse la plataforma seleccionada per a desenvolupar aquestes funcionalitats (mitjançant el llenguatge de programació JavaScript), s'ha dissenyat un únic mòdul dedicat als *triggers* de la base de dades: **TriggeredValidations.js**.

A més a més, s'han reservat dos mòduls (**GameValidation.js** i **TurnStatus.js**) per a allotjar funcions auxiliars per realitzar diferents validacions. El primer d'ells, realitza validacions de restriccions impossibles de realitzar a nivell de base de dades relacional, com pot ser la restricció d'una única partida simultània entre dos jugadors. El segon mòdul ofereix mètodes que permeten modificar i consultar l'estat d'una partida (tal i com es defineix a la màquina d'estats del joc).

Per últim, el mòdul principal que engloba els ja mencionats: **Main.js**. Aquest mòdul s'encarrega de definir els serveis web que es consultaran des de les diverses aplicacions client, així com funcions privades i auxiliars per a garantir una escalabilitat i comprensió del codi. Es pot observar un diagrama d'aquests mòduls a la Figura 34.

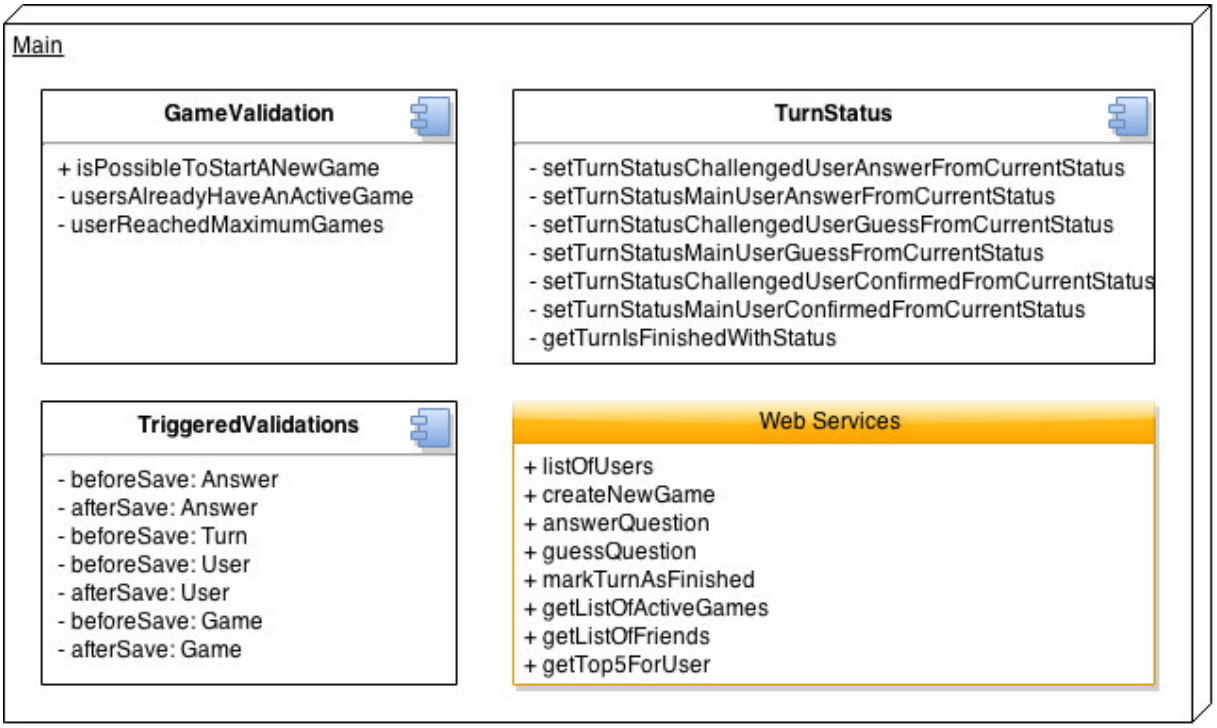


Figura 34. Mòduls de programari al servidor

Tot i que els detalls de la implementació d'aquests mòduls es detalla en una secció posterior, al diagrama superior es poden observar els principals serveis web desenvolupats, així com els mètodes auxiliars de cadascun dels mòduls esmentats.

Tots quatre mòduls es materialitzen en quatre fitxers diferents, allotjats a la carpeta *cloud* del projecte de codi al núvol específic per Parse.

FLUX D'EXECUCIÓ

Arribat aquest punt i a falta de més detall sobre el codi implementat, podem assumir que hi ha quatre entrades importants per les quals el programari al núvol pot ser executat: *createNewGame, answerQuestion, guessQuestion* i *markTurnAsFinished.* Aquests quatre punts d'entrada (serveis web) seran els encarregats de gestionar la comunicació principal entre els clients i el servidor, tal i com s'ha exposat a la Figura 32.

Més concretament, el primer d'ells gestionarà l'arribada d'una nova partida al servidor, mentre els tres següents gestionaran els estats A, G i F de la partida. Alhora que obtenen la informació pertinent del programari client, consulten els diferents mòduls de validació per tal de garantir la consistència del sistema.

Per donar més seguretat, robustesa i funcionalitats al servidor, el mòdul de *triggers* s'invoca automàticament quan es realitzen modificacions sobre entitats com *Game*, *Answer* i *Turn*, què son entitats directament tractades en els serveis web mencionats.

Al marge d'aquest flux d'execució que ateny a la partida i la seva màquina d'estats, existeixen altres serveis web necessaris per a facilitar informació al client: *getListOfActiveGames*, que obtindrà les partides actives d'un usuari; *getListOfFriends,* que retornarà els amics d'un usuari registrats al joc; i *getTop5ForUser*, que permetrà mostrar al programari client, i per cada usuari, els cinc jugadors amb major afinitat amb aquest.

CLIENT (COCOS2DX)

En el cas de les aplicacions client (Android i iOS), el disseny de programari és equivalent en totes dues plataformes, amb algunes excepcions que es mencionen posteriorment en les respectives subseccions.

A diferència de la part servidora, les aplicacions clients no utilitzen cap base de dades local ni cap plataforma de gestió automàtica de dades (com Parse), amb la qual cosa es fa necessari dissenyar un programari basat en el patró MVC (Model, Vista i Controlador) per al codi desenvolupat en C++ i Cocos2dx.

Seguit el patró mencionat, el Model emmagatzemarà les dades obtingudes de la base de dades allotjada al núvol, a través de les eines que Parse ofereix. El mòdul Vista contindrà tots els elements de la interfície gràfica, així com les diferents pantalles on ocorrerà l'acció del joc, i per últim, el mòdul Controlador gestionarà totes les comunicacions entre ambdós mòduls anteriors i diverses funcions auxiliars necessàries.

A més a més dels mòduls mencionats, existeix un darrer mòdul (*Commons Framework*) que contindrà components de programari identificats com a comuns entre les diferents tipologies de videojocs mòbils. Aquest mòdul s'ha dissenyat per implementar diverses funcionalitats que puguin ser reutilitzades en futurs desenvolupaments, tals com accés a recursos gràfics, eines de traducció de literals, eines de comunicació entre Cocos2dx i les diferents plataformes (Android i iOS), eines d'integració de publicitat als videojocs, accés als recursos de so i eines de seguiment d'ús de l'aplicació per tal de poder conèixer com els usuaris fan servir el joc.

A la Figura 35 es poden observar els quatre mòduls de que consta Preguntas Incómodas en la part client. El primer de tots, el Model, és el més senzill, ja que contindrà les entitats de la base de dades (detallada en la secció anterior), i un component per traduir fitxers en format JSON a les corresponents entitats (**Mapper**). Aquest darrer component és necessari ja que les respostes dels serveis web implementats en Parse seran en format JSON.

El segon dels components, el Controlador, consta de tres submòduls:

- **DataHolder** que contindrà, emmagatzemades en memòria, les entitats necessàries al llarg de tota l'execució del joc a la part client, com són l'usuari, les partides actives i el llistat d'amics de l'usuari.

- **ErrorHandler** és un component dedicat per llegir i informar a l'usuari final si cap error ocorre amb la comunicació client-servidor.

- **Constants**, un component amb definicions de constants que es fan servir al llarg del joc, tals com noms de fitxers d'elements gràfics, colors, etc.

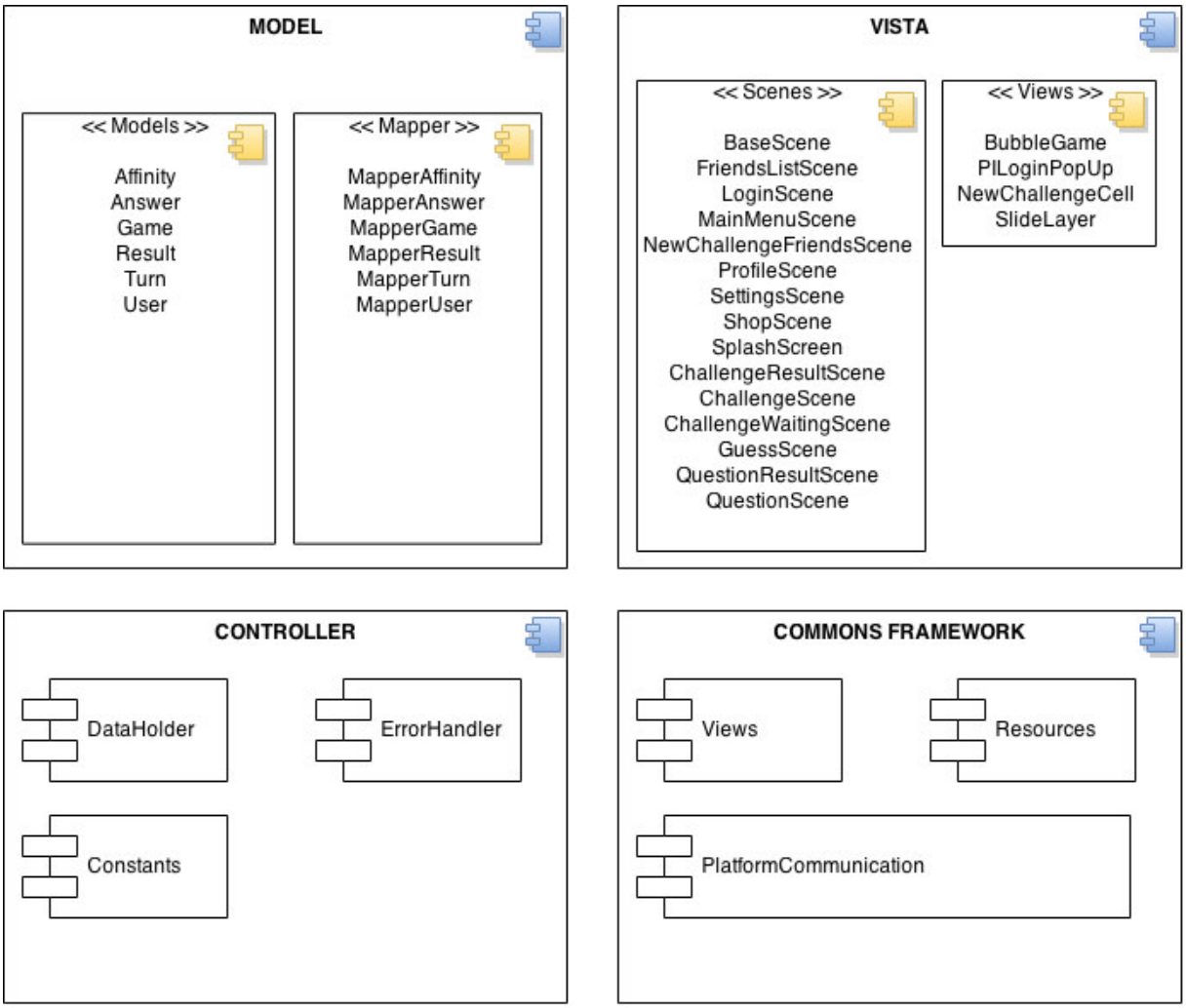


Figura 35. Disseny de software client

El tercer dels components que correspon al patró MVC, el de Vista, contindrà totes les escenes del joc i les vistes implementades per tal de facilitar el desenvolupament i la reutilització.

El llistat d'escenes (**Scenes**) es correspon amb el disseny gràfic realitzat per a l'aplicació (veure captures de pantalla de l'Annex II), mentre que les vistes (**Views**) són elements que es repeteixen en una o diverses escenes, i que s'han identificat al llarg de la implementació del projecte.

Per últim, el mòdul d'elements comuns a les diverses tipologies de jocs mòbils, *Commons Framework*, permet la comunicació entre els llenguatges de programació C++, Java i Objective-c. Aquest component conté codi en tots tres llenguatges per tal de garantir aquesta funcionalitat.

Un altre element important dintre del component Vista, és el de les vistes comuns, ja que a les aplicacions mòbils es fan servir de forma assídua elements com finestres flotants, missatges d'informació a l'usuari, etc... Totes aquestes vistes estan implementades, en la seva forma més genèrica i desacoblada, en aquest component, de tal manera que es poden reutilitzar en qualsevol aplicació i/o joc.

L'últim component de *Commons Framework* és el de recursos, que ofereix eines de localització de literals, per tal de donar suport al joc en diversos idiomes (tot i que inicialment Preguntas Incómodas només estarà disponible en espanyol). A més a més, disposarà d'eines per a l'accés i la gestió de recursos d'àudio.

CLIENT (ANDROID I IOS)

Tots els components detallats anteriorment, per sí sols, no són capaços de realitzar cap comunicació entre l'aplicació al núvol i l'aplicació client. Per això ha estat necessari desenvolupar el mòdul de comunicació entre les plataformes natives (Android i iOS) i Cocos2dx. Mitjançant aquesta comunicació, l'aplicació client pot nodrir-se de les dades oferides per les llibreries de tercers, com ara Parse, Facebook, Google Analytics o MoPub (plataforma per a la gestió de publicitat online).

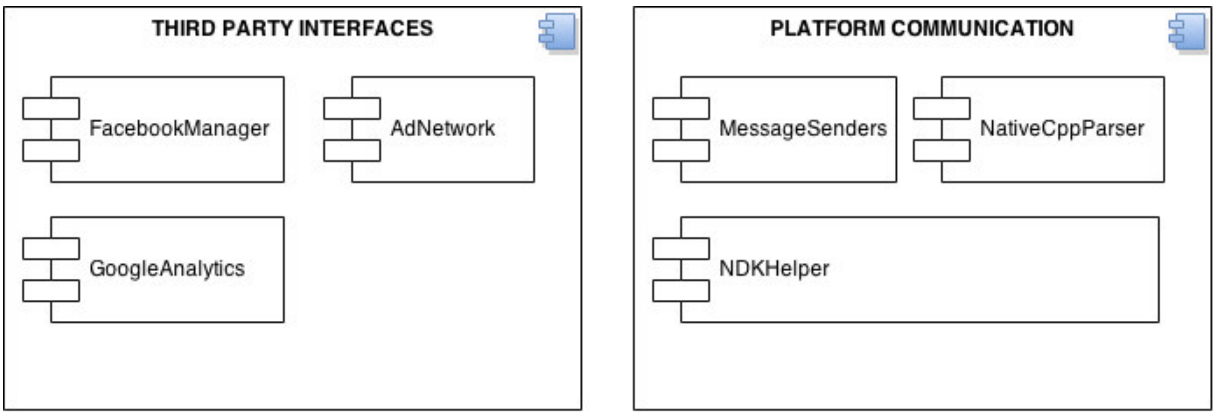


Figura 36. Components client Android

A la Figura 36 s'ofereix una visió global sobre els elements de programari dissenyats per a la plataforma Android. El primer d'ells: **Third Party Interfaces** conté tot el codi necessari per fer servir les llibreries de Google Analytics (permet obtenir estadístiques sobre l'ús que fan els usuaris de l'aplicatiu), Facebook (és indispensable per registrar-se al sistema) i la plataforma de publicitat, que mostrarà anuncis durant l'experiència de joc. Aquesta publicitat forma part del model de negoci ideat per a la rendibilitat del projecte.

D'altra banda, el component **Platform Communication** conté la interfície que connecta C++ i Java, NDKHelper [17]; un component per enviar missatges determinats des de Java a C++, MessageSenders; i un conversor de models de Parse a models de Cocos2dx, NativeCppParser. Més concretament, aquest últim component converteix un objecte de Parse en un objecte representat en format JSON, per tal de llegir-lo al codi C++.

Degut a l'ús de Cocos2dx i tots els components esmentats, es fa necessari crear un punt central de comunicació entre tots ells: **AppActivity**. Aquesta *activity* (component d'Android), rebrà tots els missatges enviats des de C++, executarà les funcions necessàries (bé de Parse, Facebook, Google Analytics o pròpies), i retornarà un missatge en format JSON a C++.

En el cas de la plataforma iOS, tots aquests components son idèntics, tot i que el punt d'encontre de totes les llibreries s'anomena **AppController**, que és el component homòleg a la *activity* d'Android a la plataforma d'Apple.

IMPLEMENTACIÓ DELS COMPONENTS DE PROGRAMARI

Un cop explicats i descrits els components de programari dissenyats per a Preguntas Incómodas, en aquesta secció s'explica la implementació d'alguns d'ells, així com curiositats i dificultats trobades al llarg del desenvolupament del projecte.

SERVIDOR (JAVASCRIPT)

La implementació de l'aplicatiu al núvol està dividit en quatre fitxers javascript:

1. **GameValidation.js**, que conté funcions javascript per realitzar comprovacions sobre la integritat de les dades al núvol.

2. **Main.js**, que és el fitxer principal del projecte, i conté la definició de serveis web que es consultaran des de les aplicacions client.

3. **TriggeredValidation.js**, conté la definició de totes les funcions que s'executen en resposta a qualsevol canvi en la base de dades, com, per exemple, la creació d'una nova partida, la modificació de l'estat d'una partida actual, etc...

4. **TurnStatus.js** és un fitxer d'ajuda que modifica l'estat d'una partida segons la definició. Aquest fitxer s'encarrega de mantenir de forma coherent la representació real de la màquina d'estats del joc, definida anteriorment en aquesta memòria.

Un exemple d'un servei web ubicat a l'aplicació al núvol és el que es mostra a la Figura 37, i que permet obtenir a qui consulti el llistat de partides actives d'un usuari. Concretament, després de cercar les partides actives d'un usuari (*main_user*) les retorna mitjançant l'entitat *response*. D'aquesta manera, qualsevol sistema autoritzat que realitzi una petició HTTP sobre aquest servei web obtindrà el llistat de partides actives entre els dos usuaris especificats.

```
/*
    List of active games for user
    Must receive user who ask

*/
Parse.Cloud.define("getListOfActiveGames", function(request, response)
{
    var main_user = request.user;

    // Montamos la query
    var gameQueryUserIsMain = new Parse.Query(Game);
    gameQueryUserIsMain.equalTo("main_user", main_user);
    gameQueryUserIsMain.notEqualTo("game_status", -1);

    var gameQueryUserIsChall = new Parse.Query(Game);
    gameQueryUserIsChall.equalTo("challenged_user", main_user);
    gameQueryUserIsChall.notEqualTo("game_status", -1);

    var mainQuery = Parse.Query.or(gameQueryUserIsMain, gameQueryUserIsChall);
    mainQuery.include("turns.question");
    mainQuery.include("main_user");
    mainQuery.include("challenged_user");
    mainQuery.ascending("createdAt");
    mainQuery.find().then(function(games){
        // for(var i=0; i<games.length ; i++){
        //
        // }
        response.success(games);
    }, function(error){
        response.error(error);
    });

});
```

Figura 37. Servei web per obtenir el llistat de partides actives

Pel que fa a les funcions que s'executen quan un objecte de la base de dades s'ha modificat (definides al fitxer TriggerValidations.js), un exemple seria l'enviament de notificacions push a tots els amics d'un usuari acabat de registrar (Figura 38).

El procés a seguir per a executar aquest tipus de funcions consisteix en definir la funció sobre l'objecte *Parse.Cloud*, com s'ha vist a la secció de fonaments teòrics sobre Parse. En aquest exemple concret, es defineix una funció a executar immediatament després que un nou usuari s'ha desat a la base de dades.

Quan aquesta acció ocorre, l'aplicació al núvol cerca a la base de dades tots els usuaris ja registrats al joc que són amics del nou usuari, i els hi envia una notificació push (que cada usuari veurà al seu dispositiu), informant-los que aquell amic s'acaba de registrar, i que ja poden jugar plegats.

En aquest cas, el missatge enviat juntament amb la notificació no està localitzat, ja que el sistema de traducció de literals a l'aplicació al núvol és molt complex i l'usuari objectiu del joc, pel moment, és de parla castellana. Es deixa, llavors, com a treball futur.

```
// - Si es un nuevo usuario enviar push a los amigos
Parse.Cloud.afterSave(Parse.User, function(request)
{
  var user = request.object;

  if(user.get("facebook_id") &&  user.get("isNew") && user.get("fb_friends").length) {
    // - User is new
    user.set("isNew", false);
    user.save().then(function(user){

        // - Ask the database for the current user
        var userQuery = new Parse.Query(Parse.User);
        userQuery.containedIn("facebook_id", user.get("fb_friends"));

        userQuery.find().then(function(results){

        var query = new Parse.Query(Parse.Installation);

        // - Para cada usuario amigo encontrado, enviar notificación push
        query.containedIn('user', results);
        Parse.Push.send(
        {
            where: query,
            data: {
                alert: "Tu amigo "+user.get("name")+" se ha unido a Preguntas Incomodas"
            }
        },
        {
            success: function() {
                console.warn("Éxito al enviar push a "+results.length+" amigos");
            },
            error: function(error) {
                console.warn("Error al enviar push a amigos");
            }
        });

    }, function(error) {
        console.warn(error);
    });
});
```

Figura 38. Trigger per a enviar notificacions push als amics d'un nou usuari

Deixant de banda el codi javascript, que queda representat pràcticament en la seva totalitat amb els dos exemples esmentatas, a continuació es mostra una captura de les preguntes introduïdes a la base de dades i que nodreixen el joc. Cal remarcar que, per tal d'afegir una nova pregunta, només cal escriure-la en el taulell de configuració de Parse i automàticament estarà disponible per a tots els jugadors.

objectId String	question_numbe…	question_string String	typology Number	createdAt Date
FZHVX67Ajd	82	¿Desconectas el móvil cuando tienes sexo?	0	Oct 15, 2014, 10:44
CRmQLTA4iO	81	¿Te llevas el móvil al baño para hacer tus necesid…	0	Oct 15, 2014, 10:44
x11pBkkJez	80	¿Has grabado un "sextape"?	0	Oct 06, 2014, 22:15
xifIFjnABL	79	¿Has embozado el WC en casa de un amigo?	0	Sep 19, 2014, 12:14

Figura 39. Preguntes disponibles a la base de dades al núvol

A l'exemple de visualització de la base de dades al núvol (Figura 39) s'observen els camps que formen la taula *Questions*. Quan algún usuari comença una partida, l'aplicatiu al núvol selecciona, de forma aleatòria, una d'aquestes preguntes per tal que els dos participants la responguin.

Un altre exemple de les dades emmagatzemades al núvol és la taula *turn*, que conté tots els torns de totes les partides (Figura 40).

objectId String	answers Array	game Pointer<Game>	turn_status_bitset String
rS5oGHCfuG	[{"__type":"Pointer","className":"Answer","objectId":"NWCUjRiWIb…	onRdkECnaE	100000
jrKvI8KmIt	[{"__type":"Pointer","className":"Answer","objectId":"uhQ5xpNG8T…	PrbJeZwKWs	110100
Beo3nQdicH	[{"__type":"Pointer","className":"Answer","objectId":"KM1LtBVvGy…	THR5X9WwUZ	111111

Figura 40. Taula 'turn' de la base de dades

D'aquesta última taula és interessant la columna *answers*, que conté una col·lecció amb totes les respostes pertanyents al torn concret. A més a més, la columna *turn_status_bitset* informa de l'estat de la partida, tal i com s'ha definit amb anterioritat en aquesta memòria.

CLIENT (COCOS2DX)

Un exemple del funcionament bàsic de tot el sistema desenvolupat es pot trobar a la pantalla d'accés a l'aplicació (Figura 41).


```
// - Main title
Sprite* logo = Sprite::create(kLogo);
logo->setAnchorPoint(Vec2(0.5f, 0.0f));
logo->setPosition(middle.x, 3*MDVisibleRect::getVisibleRect().size.height/4);
addChild(logo);
```

```
// - Top background
Sprite* topBg = Sprite::create(kBgTop);
topBg->setScale(MDVisibleRect::getVisibleRect().size.width/topBg->getContentSize().width);
topBg->setAnchorPoint(Vec2(0.5, 1.0f));
topBg->setPosition(middle.x, MDVisibleRect::getVisibleRect().size.height);
addChild(topBg);
```

```
// - Login with Facebook button
MenuItemImage* btnLogin = MenuItemImage::create(kBtnLoginDefault, kBtnLoginPressed);
btnLogin->setPosition(middle);
btnLogin->setCallback([this](Ref* sender){
    MRGoogleAnalyticsFactory::NativeTrackEvent("action", "login",
                                               "Click on Login with Facebook from Login Screen");

    showLoading();

    // - Send a message to the native code of the client application
    SendMessageWithParams("doFacebookLogin", NULL);
});

// - Login with facebook button title
LabelTTF *title = LabelTTF::create(MDStringManager::getInstance()
                                   ->getLocalizedString("login_with_facebook"),
                                   RALEWAY_SEMIBOLD, kLabelSmall);
title->setColor(Color3B(255, 255, 255));
title->setPosition(btnLogin->getContentSize().width/2, btnLogin->getContentSize().height/2);

// - Add the button title to the button
btnLogin->addChild(title);
```

Figura 41. Exemple de codi C++ a Cocos2dx

Al codi de la figura s'observa el procés de creació de dos elements gràfics: el fons de la zona superior de la pantalla i el títol principal. Ambdós elements son de tipus *Sprite*, objecte que es construeix amb el nom del fitxer gràfic que contindrà (normalment un fitxer PNG). En aquest exemple, *kBgTop* i *kLogo* són constants, definides al fitxer *ResourcesConstants.hpp,* i de tipus cadena de caràcters, on s'especifica la ruta de cada fitxer PNG.

Cal destacar també la creació del botó amb el títol "Acceder con Facebook", ja que presenta diversos conceptes interessants:

- És un component de Cocos2dx anomenat *MenuImageButton*, i es construeix a partir de dues imatges PNG: una per a l'estat normal (sense prémer), i un altre per a l'estat premut.

- A aquest botó se li afegeix un títol, el text del qual està localitzat en diversos idiomes i s'obté a través del component *MDStringManager*, implementat al component *commons framework* ja esmentat en la secció de disseny de programari.

- La funcionalitat del botó quan l'usuari el prem es defineix mitjançant la directiva *setCallback* i especificant un fragment de codi immediatament després. Aquest codi mereix un especial èmfasi:

```
btnLogin->setCallback([this](Ref* sender){
    MRGoogleAnalyticsFactory::NativeTrackEvent("action", "login",
                                               "Click on Login with Facebook from Login Screen");

    showLoading();

    // - Send a message to the native code of the client application
    SendMessageWithParams("doFacebookLogin", NULL);
});
```

Figura 42. Codi de comportament del botó "Acceder con Facebook"

a) La primera de les instruccions (*MRGoogleAnalyticsFactory*) envia un missatge a la part nativa de l'aplicatiu (Android o iOS), que enviarà l'esdeveniment "*Click on Login with Facebook from Login Screen*" a Google Analytics a través de la llibreria que Google ofereix per Android i iOS, amb la qual cosa es pot obtenir un seguiment del comportament de l'usuari dintre del joc.

b) La segona instrucció, *showLoading*, mostrarà un component gràfic per tal d'informar a l'usuari que s'està realitzant una tasca asíncrona de connectivitat amb l'aplicatiu al núvol. Aquesta instrucció és possible gràcies a la jerarquia d'escenes implementada: existeix una escena bàsica (BaseScene), de la qual hereten la resta d'escenes, i que conté una sèrie de mètodes que poden ser utilitzats per totes les escenes, com la mostra d'un component gràfic per indicar l'execució d'una tasca asíncrona, la mostra d'un missatge d'error o la gestió de possibles interaccions de l'usuari amb certs botons.

c) Pel que fa a la última instrucció i, tal i com indica el comentari que l'encapçala, s'envia un missatge al codi nadiu de la plataforma determinada, per tal que es llenci el procés de registre (o accés) de l'usuari a través de la llibreria de Facebook i Parse.

Un cop l'usuari ha accedit a l'aplicació, és a dir, s'ha identificat a través de Facebook, la següent pantalla que es mostra és l'escena de les partides actives en aquell instant de temps.

Existeixen dos mètodes de cada objecte *Scene* que controlen el flux d'aparició i desaparició de l'escena de la pantalla: *onEnter* i *onExit.* Aquests mètodes prenen especial importància en l'escena de les partides actives, ja que han de consultar al núvol les dades de l'usuari i les seves partides actives.

```
void MainMenuScene::onEnter() {
    BaseScene::onEnter();

    // - Active Challenges List
    NDKHelper::AddSelector("MainMenuActiveGames", "retrieveActiveGamesSuccess",
                           callfuncND_selector(MainMenuScene::retrieveActiveGamesSuccess),
                           this);

    NDKHelper::AddSelector("MainMenuOnError", "onError",
                           callfuncND_selector(BaseScene::onError),
                           this);
    NDKHelper::AddSelector("SetupCurrentUserMainMenu", "updateCurrentUser",
                           callfuncND_selector(MainMenuScene::updateCurrentUser),
                           this);
    SendMessageWithParams("updateCurrentUser", NULL);
    SendMessageWithParams("getListOfActiveGames", NULL);
}


void MainMenuScene::onExit() {

    // - Active Challenges List
    NDKHelper::RemoveSelectorsInGroup("MainMenuActiveGames");
    NDKHelper::RemoveSelectorsInGroup("MainMenuOnError");
    NDKHelper::RemoveSelectorsInGroup("SetupCurrentUserMainMenu");

    BaseScene::onExit();
}
```

Figura 43. Mètodes onEnter i onExit de MainMenuScene

La plataforma Cocos2dx fa una crida al mètode *onEnter* quan aquesta escena es mostra en pantalla, i és llavors quan es realitzen dues operacions fonamentals per al funcionament de la comunicació entre C++ i Java (o Objective-c):

1. Es registren els mètodes que s'executaran quan el codi nadiu (Java o Objective-c) enviï un missatge concret, com per exemple *onError, updateCurrentUser* o *retrieveActiveGamesSuccess.* Es pot dir que aquests mètodes son els receptors del resultat de la comunicació client-servidor.

2. Seguidament, s'envia un missatge al codi nadiu per tal que realitzi dues peticions al servidor. Una per obtenir les dades de l'usuari actual i una altra per obtenir el llistat de partides actives en aquell instant precís.

És en aquest moment quan el control del programa es troba a la part Java (o Objective-c). Per exemple, a la Figura 44 es mostra el fragment de codi Java que realitza la petició, mitjançant Parse, per obtenir el llistat de partides actives de l'usuari actiu.

L'objecte *ParseCloud* ens permet realitzar una crida al codi del núvol, i ens notifica quan el resultat de la comunicació està disponible, amb una col·lecció d'objectes o un error (en cas que hi hagi). Si no hi ha hagut cap error, s'utilitza la classe *MainMenuSceneCppMessageSender* per enviar els resultats al codi C++. Aquesta classe pertany al component *MessageSenders* (Figura 36) esmentant a la secció de disseny de programari, i el seu objectiu no és més que transformar els objectes obtinguts de la comunicació client-servidor, convertir-los a format JSON i enviar-los de nou al codi C++.

```java
/**
 * Gets the list of active games from the user
 *
 * @param object The JSONObject sent from C++ (not used but required)
 */
public void getListOfActiveGames(JSONObject object) {

    String userId = ParseUser.getCurrentUser().getObjectId();

    HashMap<String, Object> params = new HashMap<String, Object>();
    params.put("main_user_id", userId);

    ParseCloud.callFunctionInBackground("getListOfActiveGames", params, new FunctionCallback<List<ParseObject>>() {

        public void done(List<ParseObject> results, ParseException e) {
            if(e == null) {
                // - Success!
                MainMenuSceneCppMessageSender.getInstance().sendActiveGames(results);
            } else {
                // - TODO: Handle error
                MainMenuSceneCppMessageSender.getInstance().sendError(e.getMessage());
            }
        }
    });
}
```

Figura 44. Fragment Java per obtenir el llistat de partides actives

```java
public class MainMenuSceneCppMessageSender extends CppMessageSender {

    protected static MainMenuSceneCppMessageSender instance = null;

    public static MainMenuSceneCppMessageSender getInstance() {
        if (instance == null) {
            instance = new MainMenuSceneCppMessageSender();
        }
        return instance;
    }

    public void sendActiveGames(List<ParseObject> objects) {
        AndroidNDKHelper.SendMessageWithParameters("retrieveActiveGamesSuccess", (JSONArray)
            ParseCppAutoConverter.ParseToNativeConverter(objects));
    }
}
```

Figura 45. Classe MainMenuSceneCppMessageSender (Java)

Cal destacar, a la classe *MainMenuSceneCppMessageSender* l'ús del component *ParseCppAutoConverter* (pertanyent al mòdul definit com *NativeCppParser*), que transforma qualsevol objecte que Parse ens ofereix a tipus JSON.

Un cop realitzada aquesta transformació, s'envia un missatge al codi C++ amb el missatge concret i els objectes desitjats:

```
AndroidNDKHelper.SendMessageWithParameters("retrieveActiveGamesSuccess",..)
```

Aquest missatge té definit un mètode receptor a l'escena *MainMenuScene* (Figura 43), amb la qual cosa ja es disposa de la informació necessària, obtinguda des del núvol, i en format JSON a la secció C++. Arribats a aquest punt, només cal fer servir les classes implementades al component *Mapper* del model i transformar els objectes JSON en objectes propis del model implementat a C++ (Figura 35).

```
void MainMenuScene::retrieveActiveGamesSuccess(Node* pSender, void* data)
{
    hideLoading();

    MapperModelGame modelGameMapper;
    GamesDataHolder::getInstance().setGamesVector(modelGameMapper.parseArray(data));
    std::vector<ModelGame> activeGames = GamesDataHolder::getInstance().getUpdatedGamesVector();
    buildChallengeBubbles(activeGames);
    scheduleUpdate();
}
```

Figura 46. Mètode receptor de les partides actives a C++

El segon dels mètodes dels que s'ha parlat (*onExit*) s'executa sempre que l'escena desapareix de la pantalla, moment en el qual s'han d'eliminar els mètodes receptors de missatges per tal de no executar-se quan l'escena no es troba en pantalla.

D'aquesta manera queda palès el procés de comunicació entre codi C++, Java i l'aplicació al núvol. Existeixen nombroses comunicacions client-servidor al llarg de tot el projecte, seguint el mateix procediment i components per garantir el correcte funcionament del joc.

FASE DE PRODUCCIÓ

Un cop el projecte està desenvolupat i ha passat les convenients fases de proves i qualitat, està llest per ser publicat a les diferents tendes d'aplicacions i observar el comportament dels usuaris.

Durant aquesta fase, de durada aproximada d'entre dues i tres setmanes, el producte es troba disponible per a la descàrrega i es realitza un exhaustiu treball de publicitat i adquisició d'usuaris. Alhora, s'obté informació sobre el rendiment i la robustesa del producte, planificant una fase de desenvolupament posterior per tal de solucionar els problemes que els usuaris hagin reportat després de l'ús de l'aplicació.

Per tant, aquesta secció es pot dividir en dues subseccions: adquisició d'usuaris i seguiment del funcionament.

ADQUISICIÓ D'USUARIS

L'adquisició d'usuaris és una de les accions més importants a realitzar durant la fase de producció d'un videojoc mòbil. Sabut és que a les tendes d'aplicacions hi ha al voltant d'un bilió d'aplicacions disponibles per a descarregar, de les quals quasi un 45% són videojocs.

D'entre tots aquests videojocs, l'usuari ha de ser capaç de trobar el nostre videojoc, descarregar-lo i continuar jugant durant els pròxims dies, setmanes o mesos, en el millor dels casos.

Per tal de fer notar Preguntas Incómodas sobre la resta d'aplicacions del mercat, s'ha realitzat una estratègia d'ASO [18] (*Application Search Optimization*) que inclou

des de notes de premsa fins a publicacions en mitjans digitals especialitzats, campanyes de xarxes socials i publicitat digital.

PREMSA DIGITAL ESPECIALITZADA

Un dels paràmetres que les tendes d'aplicacions tenen en compte per tal de donar més o menys importància a una aplicació o joc és el nombre d'enllaços que Internet disposa cap a la teva aplicació. Per tant, publicar una nota de premsa en blocs especialitzats, fòrums i premsa genèrica facilita l'adquisició d'usuaris. A l'Annex III es troba la publicació realitzada al bloc de Manduka Games (http://blog.mandukagames.com) que es va enviar a diferents mitjans de comunicació digital.

Cal destacar el llenguatge informal i atractiu per incentivar la descàrrega, així com la imatge central amb la frase "**¡Descárgala ya!**", que és un enllaç a la pàgina de descàrrega de l'aplicació.

Cap al final de la publicació s'anima als lector a participar en les xarxes socials mitjançant el *hashtag* #PreguntasIncómodas, on es demana que formulin les preguntes que els hi agradaria trobar a l'aplicació per tal d'incorporar-les a futures versions.

PUBLICITAT DIGITAL

Malgrat la bona acollida dels portals digitals especialitzats en videojocs envers la publicació de Preguntas Incómodas, el gruix de l'estratègia d'adquisició d'usuaris resideix en la publicitat digital. En aquest cas, es va realitzar una feroç campanya d'anuncis a la xarxa social Facebook, mostrant l'anunci de la Figura 47 al mur de tots aquells usuaris que complien una sèrie de característiques específiques.

Figura 47. Anunci de Facebook

Aquesta publicitat és la més utilitzada pels desenvolupadors d'aplicacions i videojocs per a dispositius mòbils avui en dia. Facebook factura una quantitat per cada cop que l'anunci ha estat vist per algun dels seus usuaris, i una altra si aquest usuari ha acabat instal·lant l'aplicació suggerida. A la secció de conclusions s'analitza la inversió realitzada per Preguntas Incómodas i els resultats obtinguts.

Una altra eina online i gratuïta que es fa servir per conèixer el funcionament d'un joc o una aplicació mòbil és App Annie [19]. Aquesta aplicació mostra, d'entre altres coses, la posició que una aplicació ocupa en la llista d'aplicacions de Google Play o Apple Store.

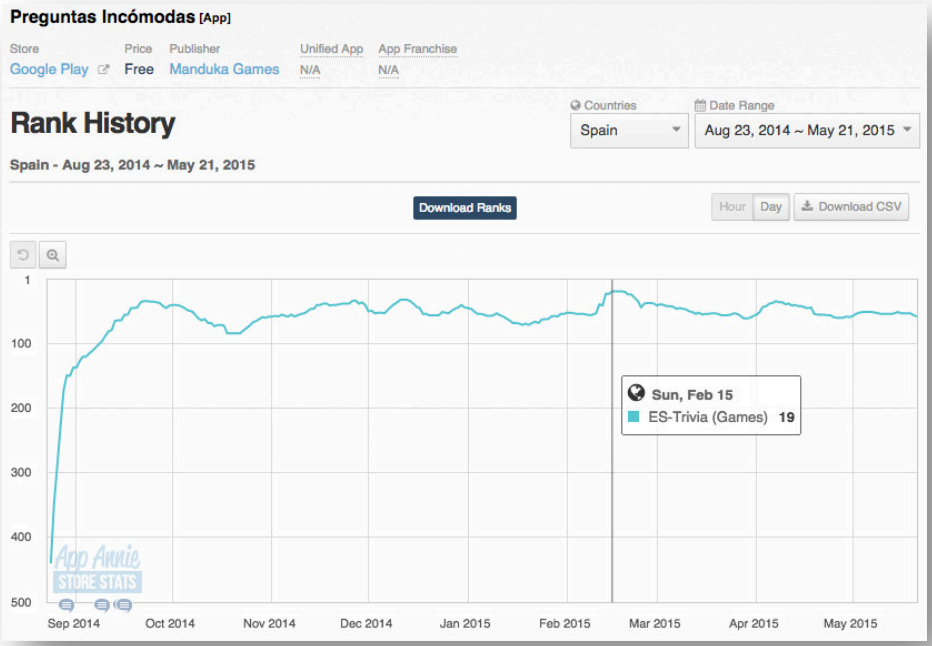


Figura 48. Posició Preguntas Incómodas en Android (categoria Trivia)

A la Figura 48 es mostra la evolució que Preguntas Incómodas ha sofert en quan a posicionament en la categoria Trivia de Google Play. La millor posició és la número 19, el que significa que qualsevol persona que busqués els millors jocs de la categoria Trivia en la tenda d'aplicacions d'Android d'Espanya, veuria Preguntas Incómodas en la posició 19.

Aquest seguiment és fonamental per tal de potenciar l'adquisició d'usuaris orgànics, és a dir, usuaris que es descarreguen l'aplicació per interès propi sense ésser incentivats.

SEGUIMENT DEL FUNCIONAMENT

Un cop coberta tota la feina a realitzar per tal d'adquirir una gran massa d'usuaris, cal detallar el seguiment que es realitza del funcionament de l'aplicació i quins paràmetres i estadístiques es consulten per conèixer com està funcionant el joc.

Tant la tenda d'aplicacions d'Apple com la d'Android ens ofereixen estadístiques sobre el nombre de descàrregues en un període de temps determinat, així com els errors que els usuaris han trobat al llarg de la seva experiència amb el joc. Aquesta última informació és realment útil per tal de detectar els errors i planificar una futura fase de desenvolupament per tal de solucionar-los.

A més a més, Parse també ens ofereix estadístiques sobre els accessos al programari al núvol, el nombre de registres que hi ha, el nombre de partides, etc...

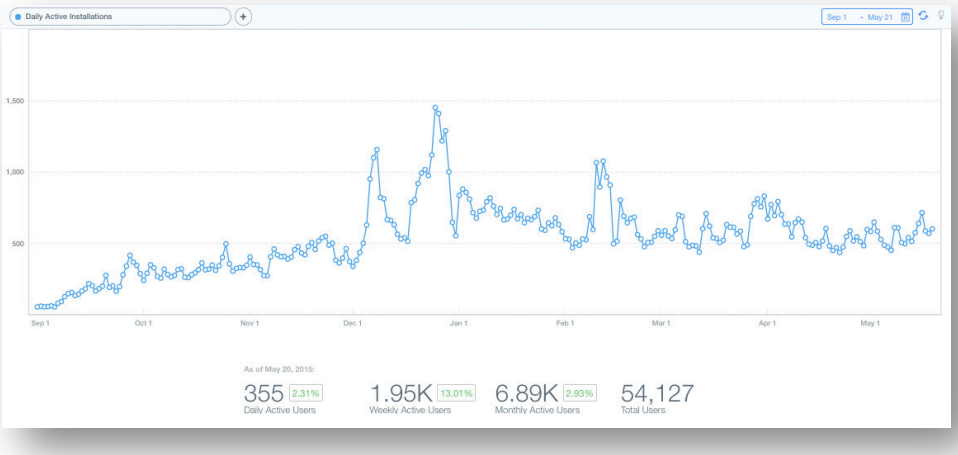


Figura 49. Estadístiques Parse

A la Figura 49 s'observen aquestes dades que Parse ens ofereix. La gràfica mostra les instal·lacions de l'aplicació que ocorren diàriament, mentre que en la zona inferior es mostren xifres com els DAU (usuaris actius diaris), els MAU (usuaris actius mensuals) i el total d'usuaris registrats.

A més a més, Parse ofereix dades sobre les quotes que l'aplicació està fent servir dintre del pla contractat (Figura 50). Aquestes dades mostren el nombre de connexions simultànies al servidor per segon o el total d'emmagatzematge fet servir per l'aplicació al núvol.

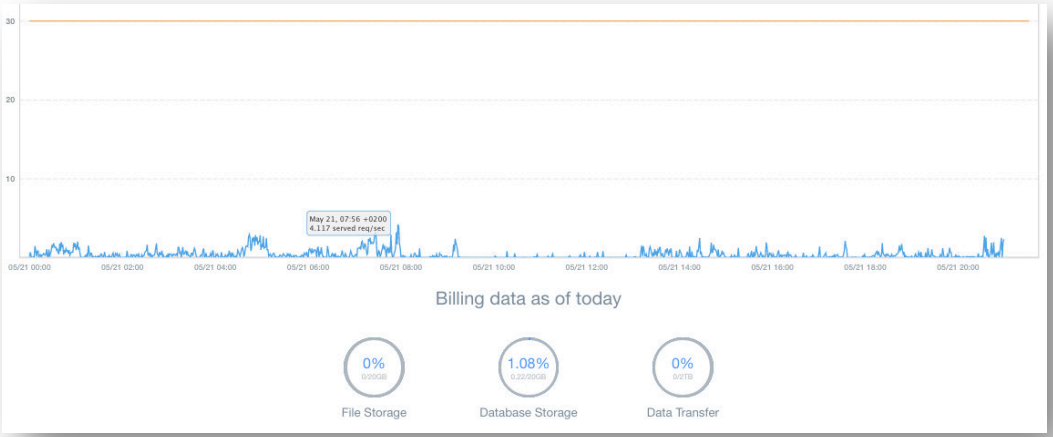


Figura 50. Dades sobre la quota de Parse

La franja taronja superior que es mostra a la gràfica de la Figura 50 és el límit de connexions simultànies per segon per al pla gratuït de Parse. Preguntas Incómodas no arriba al 15% de la quota, amb la qual cosa queda palesa la potència i funcionalitats que ofereix Parse sense cap cost.

FASE DE POSTPRODUCCIÓ

La fase de postproducció contempla tots els anàlisis realitzats després de les primeres setmanes de vida del producte, valorant l'èxit de la idea inicial, la qualitat del producte final, la seva robustesa i el comportament dels usuaris. Després d'aquest anàlisis, es poden extreure conclusions taxatives que determinaran el treball futur, és a dir, si existeixen indicadors que garanteixin l'èxit i la viabilitat del joc.

ANÀLISI

L'anàlisi de les dades, gràcies a la integració de Google Analytics i les eines de Parse, es pot dividir en dos blocs importants: la retenció dels usuaris, que mesurarà la qualitat del producte, així com la satisfacció dels usuaris, i l'anàlisi de la captació d'usuaris, en aquest cas mitjançant la publicitat digital.

ANÀLISI DE RETENCIÓ D'USUARIS

L'anàlisi sobre la retenció dels usuaris posa de manifest el grau en que els usuaris tornen a l'aplicació després del primer cop. Com és d'esperar, quants més usuaris tornin després del primer cop (i del segon, i del tercer...), millor serà la retenció, i significarà que tan la idea com el software són de qualitat i d'interès pels jugadors.

Gràcies a Parse, aquest anàlisi es pot realitzar de forma senzilla i ràpida fent una ullada a la gràfica que es mostra al tauler de gestió (Figura 51).

Figura 51. Gràfica de retenció d'usuaris a Parse

La gràfica de retenció mostra, tan en l'eix vertical com en l'horitzontal, una línia de temps relativa des del dia que es consulta. Així doncs, la gràfica de la Figura 51 mostra, en la primera de les seves caselles, que el 16'2% dels usuaris que es varen donar d'alta 28 dies enrere, accediren a l'aplicació al dia següent.

La dada que s'utilitza (per regla no escrita) per a mesurar la retenció és el percentatge d'usuaris que accedeixen a l'aplicació després de 8 dies de la seva descarrega. Pel cas de Preguntas Incómodas, aquest percentatge es mostra a la Figura 51, i té un valor de **1'4%**. Aquest indicador mostra que el cicle de vida d'un usuari al joc és molt curt i l'abandona bastant aviat. Això es pot deure a possibles errors de programari, que fan que l'experiència d'usuari no sigui l'adequada, fins a la frustració que pot generar el fet de limitar el joc a persones de la pròpia coneixença, és a dir, un usuari no podrà jugar fins que algún amic es descarregui també el joc. I tal com s'ha esmentat anteriorment, aquest anàlisi obliga a planificar un treball futur per tal d'incrementar la retenció dels usuaris i augmentar la durada de l'experiència de joc.

ANÀLISI DE PUBLICITAT DIGITAL

Pel que fa a l'anàlisi de la publicitat digital, per Preguntas Incómodas la inversió en publicitat a Facebook va ser de **377'08 USD**, amb la qual cosa l'anunci va ser vist quasi 60.000 cops, i l'aplicació va ser instal·lada 1.781 cops.

Aquestes xifres generen un CPI (cost per instal·lació) de 0'21 USD, dada que es fa servir per valorar la qualitat de la campanya d'adquisició d'usuaris i si el producte pot arribar a ser rendible o no. A la Figura 52 es mostra el gràfic d'aquesta inversió publicitària durant els darrers mesos de 2014.

Figura 52. Gràfic d'inversió en publicitat

¿Com es pot saber, durant les primeres setmanes de vida del producte, si pot arribar a ser rendible o no? A partir del **CPI**: 0'21 USD. Aquesta dada significa que adquirir un usuari té un cost de 21 cèntims però, quin **ARPU** [20] (Average Revenue Per User) estimem tindrà la nostra aplicació? Aquest ARPU és el la despesa mitja que cada usuari realitzarà sobre el model de negoci de l'aplicació.

Per exemple, jocs famosos com el *Candy Crush Saga* obtenen un ARPU de 0'75 USD, mentre que *Clash of Clans* n'obté un 1'31 USD [21]. Assumit que el CPI dels seus jocs ronda els 0'20 USD, com Preguntas Incómodas, guanyen 0'55 i 1'11 dòlars de mitja per usuari, respectivament. Aquesta és la regla d'or que garanteix que el model de negoci implementat al videojoc (o aplicació) és completament rendible, ja que un usuari realitza una despesa major del que costa adquirir-lo.

En Preguntas Incómodas el CPI ens mostra que es tracta d'un joc de bona qualitat i atractiu per als usuaris objectiu, amb la qual cosa només resta implementar un model de negoci que permeti obtenir un ARPU superior a 0'21 USD, i la rendibilitat estarà garantida.

CONCLUSIONS

Arribats a aquest punt i després d'analitzar totes les dades recol·lectades durant la fase de producció, es poden extreure les següents conclusions:

1. **Preguntas Incómodas és un bon candidat per ser un videojoc rendible a l'ecosistema mòbil i de micro pagaments**. Basant-nos en el constant creixement d'usuaris registrats (més de 50.000), els usuaris mensuals actius (prop dels 7.000) i el percentatge de conversió del sector (entre un 1 i un 2% dels usuaris actius mensuals es gasten una mitja de 3 USD diaris), es dedueix que Preguntas Incómodas podria generar **12.600 EUR** mensuals de facturació.

2. **Cocos2dx ha estat la plataforma perfecta**, permetent un desenvolupament en C++ executable a les plataformes objectiu (Android i iOS), així com la fàcil integració amb la plataforma al núvol.

3. **El més important de la fase de producció és la campanya d'adquisició d'usuaris,** ja que l'escenari de videojocs mòbils conté milers de milions d'aplicacions, i destacar sobre tots ells no és tasca senzilla.

4. En un ecosistema tan variable, **és important analitzar el comportament dels usuaris a l'aplicació i actuar en conseqüència,** garantint una bona retenció, experiència d'usuari i, en definitiva, més facturació.

TREBALL FUTUR

Havent conclòs que Preguntas Incómodas és un videojoc candidat per a obtenir beneficis al mercat dels micro pagaments mòbils, una de les tasques més importants a realitzar en el futur més immediat és la implementació d'aquest model de negoci. Aquest model de negoci requereix inserir a l'experiència de joc dos conceptes nous: **monedes** i **tokens**.

Els usuaris aniran guanyant monedes durant l'experiència de joc, i les podran gastar per adquirir **tokens** en una primera fase. Aquests tokens es podran intercanviar per comodins durant la fase de resposta d'una pregunta, permetent a l'usuari no

respondre la pregunta i, en conseqüència, no mostrar la resposta al contrincant. A més a més, fent ús d'aquest comodí, l'usuari obtindrà automàticament la màxima puntuació del torn.

La decisió d'introduir els dos conceptes en comptes d'un és per facilitar futures funcionalitats i elements comprables, com puguin ser acceleradors, la capacitat de personalitzar preguntes, etc... Els acceleradors permetrien als usuaris realitzar més accions de les permeses en un rang determinat de temps. Aquest darrer concepte es coneix com barra d'energia, que només permetria a l'usuari respondre un nombre determinat de preguntes en 24 hores (per exemple). Fent servir els acceleradors, podria respondre més preguntes de les predeterminades.

A més a més d'aquesta implementació del model de negoci, el treball futur contempla la possibilitat de crear partides de més d'un torn (com ja es preveia a la fase de disseny), concretament tres torns, amb la qual cosa l'experiència de joc és més extensa i la retenció dels usuaris major, derivant en un increment d'ingressos. I, un cop el producte hagi demostrat de manera taxativa la seva rendibilitat, caldrà una inversió futura (en termes d'esforç i econòmics) per tal d'expandir el territori objectiu del producte, és a dir, passar a donar cobertura al món sencer. Aquesta cobertura mundial requereix la implementació de un sistema de localització de literals a la part servidora, a la part client i la creació de noves campanyes de publicitat lligades als diferents països objectiu.

Deixant de banda les funcionalitats d'expansió del projecte, s'ha de reservar una bossa d'hores per solucionar els errors de l'aplicació apareguts durant les primeres setmanes de vida del producte. De ben segur que, quan tot aquest treball futur estigui finalitzat, Preguntas Incómodas serà un dels projectes punters en l'expoltació del model de negoci basat en micro pagaments, tal i com es pretenia des del començament.

Bibliografia

[1] eMarketer. (2015, May) Smartphone Users Worldwide Will Total 1.75 Billion in 2014. [Online].
http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536

[2] eMarketer. (2015, May) In-App Purchases Take Over App Revenues. [Online].
http://www.emarketer.com/Article/In-App-Purchases-Take-Over-App-Revenues/1010491

[3] Parse. (2015, Mar.) Parse. [Online].
https://parse.com/

[4] Apple. (2015, Mar.) XCode IDE. [Online].
https://developer.apple.com/xcode/ide/

[5] Eclipse.org. (2015, Mar.) Eclipse IDE for Java Developers. [Online].
https://eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr2

[6] Google. (2015, Mar.) Android Developer Tools. [Online].
http://developer.android.com/tools/help/adt.html

[8] Apple. (2015, Mar.) Objective C.

[7] Google. (2015, Mar.) Android NDK. [Online].
http://developer.android.com/tools/sdk/ndk/index.html

[9] Oracle. (2015, Mar.) Java. [Online].
http://java.com/en/

[10] cplusplus. (2015, Mar.) C++ Resources Network. [Online].
http://www.cplusplus.com/

[11] Oracle. (2015, Apr.) Database Triggers. [Online].
http://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch15.htm

[12] Chukong Technologies. (2015, Mar.) Cocos2d-x. [Online].
http://www.cocos2d-x.org/about

[13] Chukong Inc. (2015, May) Coco Studio. [Online].
http://www.cocos2d-x.org/wiki/Cocos_Studio

[14] Depto. Ciencia de la Computación e IA. (2015, May) Moodle UA. [Online].
https://moodle2014-15.ua.es/moodle/pluginfile.php/18330/mod_resource/content/2/vii-03-cocos2dx.pdf

[15] DigiCert. (2015, May) What is SSL. [Online].
https://www.digicert.com/ssl.htm

[16] Agencia Española de Protección de Datos. (2015, May) Reglamento de la LOPD. [Online].
https://www.agpd.es/portalwebAGPD/canaldocumentacion/informes_juridicos/reglamento_lopd/index-ides-idphp.php

[18] Tribal Worldwide Spain. (2015, May) IAB Spain. [Online].
http://www.iabspain.net/wp-content/uploads/downloads/2014/09/Manual-ASO-2014.pdf

[17] Amir Ali Jiwani. (2015, May) GitHub - EasyNDK. [Online].
https://github.com/aajiwani/EasyNDK-for-cocos2dx

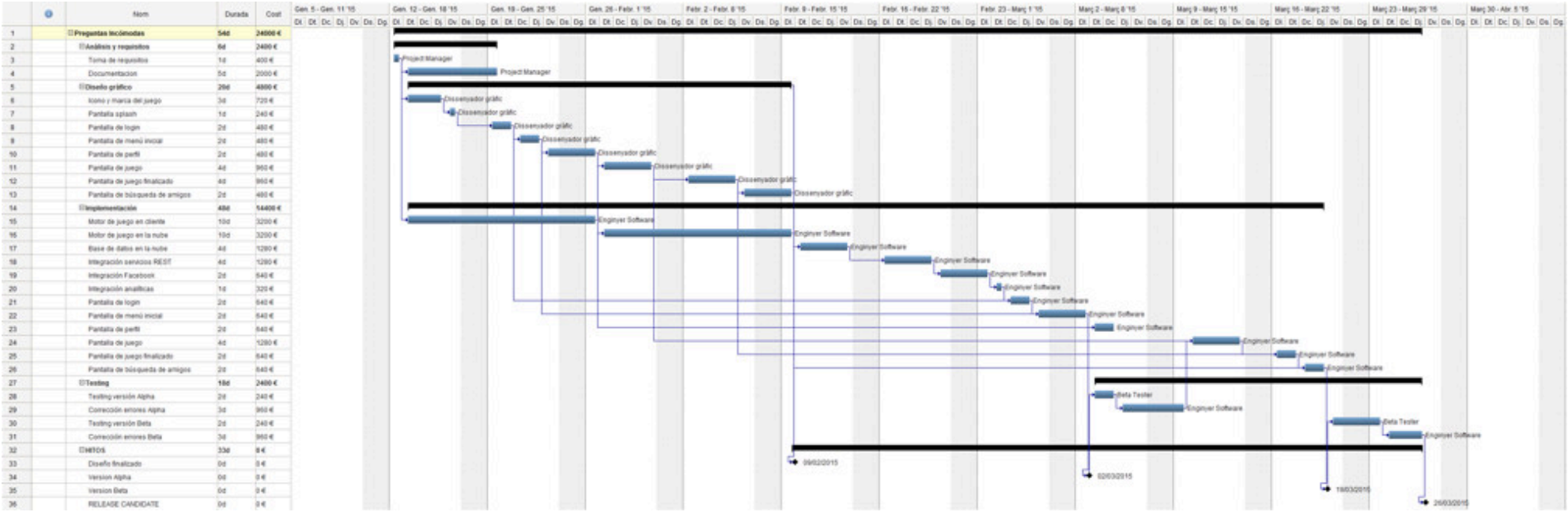
[19] App Annie. (2015, May) App Annie. [Online].
https://www.appannie.com/

[20] Ernesto Piedras. (2015, May) El Economista. [Online].
http://eleconomista.com.mx/columnas/columna-especial-empresas/2012/06/27/arpu-movil

[21] Statista. (2015, May) Statista. [Online].
http://www.statista.com/statistics/346412/arpu-mobile-games/

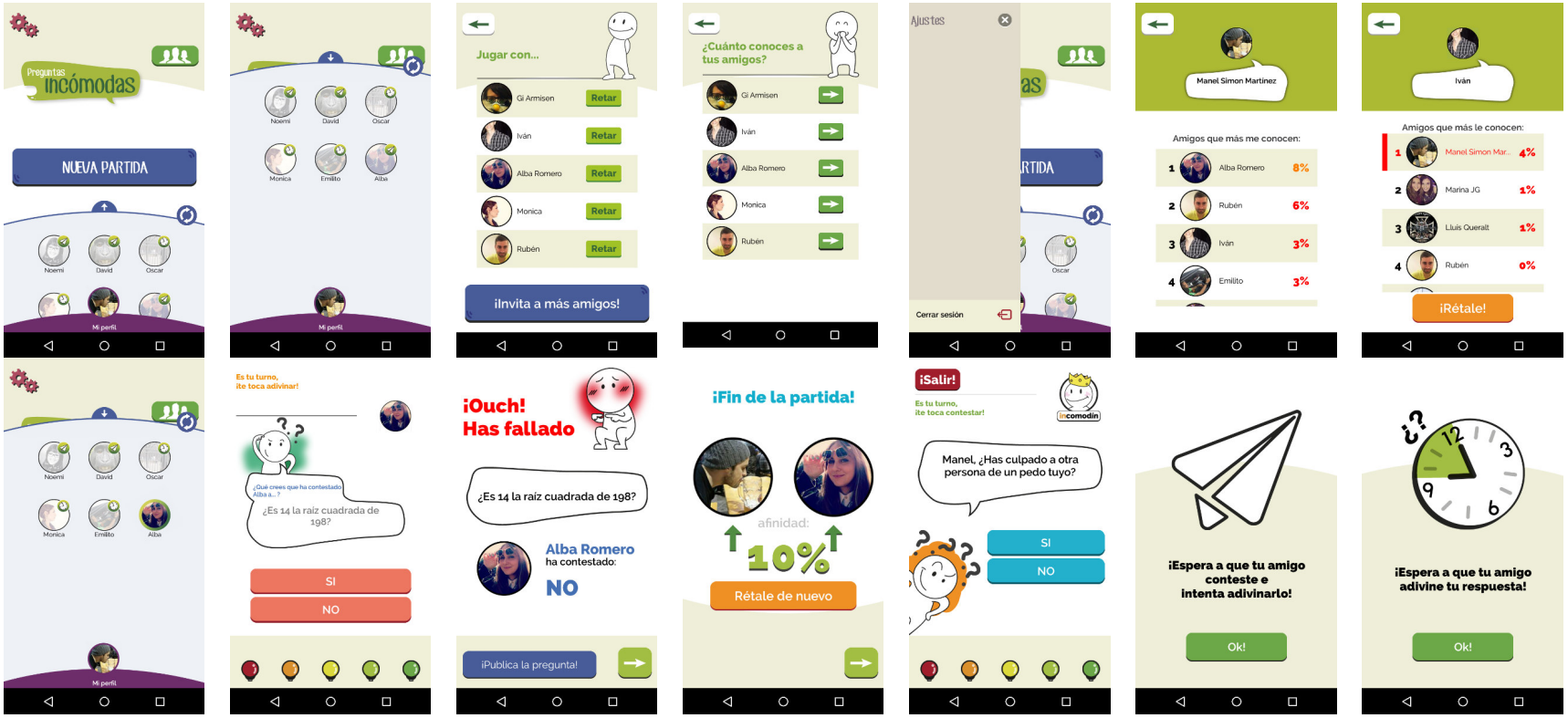
[22] Hasbro. (2009, Jan.) Hasbro. [Online].
http://www.hasbro.com/common/documents/dad2af521c4311ddbd0b0800200c9a66/ADFE82BD50569047F59AFE46C03986C7.pdf

Annex I. Diagrama de Gantt - Planificació Inicial



	Nom	Durada	Cost
1	**Preguntas incómodas**	**54d**	**24000 €**
2	**Análisis y requisitos**	**6d**	**2400 €**
3	Toma de requisitos	1d	400 €
4	Documentacion	5d	2000 €
5	**Diseño gráfico**	**20d**	**4800 €**
6	Icono y marca del juego	3d	720 €
7	Pantalla splash	1d	240 €
8	Pantalla de login	2d	480 €
9	Pantalla de menú inicial	2d	480 €
10	Pantalla de perfil	2d	480 €
11	Pantalla de juego	4d	960 €
12	Pantalla de juego finalizado	4d	960 €
13	Pantalla de búsqueda de amigos	2d	480 €
14	**Implementación**	**48d**	**14400 €**
15	Motor de juego en cliente	10d	3200 €
16	Motor de juego en la nube	10d	3200 €
17	Base de datos en la nube	4d	1280 €
18	Integración servicios REST	4d	1280 €
19	Integración Facebook	2d	640 €
20	Integración analíticas	1d	320 €
21	Pantalla de login	2d	640 €
22	Pantalla de menú inicial	2d	640 €
23	Pantalla de perfil	2d	640 €
24	Pantalla de juego	4d	1280 €
25	Pantalla de juego finalizado	2d	640 €
26	Pantalla de búsqueda de amigos	2d	640 €
27	**Testing**	**15d**	**2400 €**
28	Testing versión Alpha	2d	240 €
29	Corrección errores Alpha	3d	960 €
30	Testing versión Beta	2d	240 €
31	Corrección errores Beta	3d	960 €
32	**HITOS**	**33d**	**0 €**
33	Diseño finalizado	0d	0 €
34	Version Alpha	0d	0 €
35	Version Beta	0d	0 €
36	RELEASE CANDIDATE	0d	0 €

Annex II. Captures de pantalla



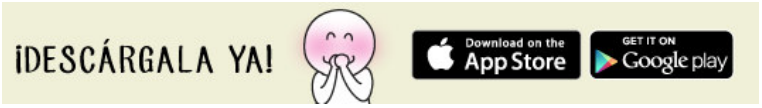
Annex III. Blog digital i nota de premsa

¿Te atreves con el juego de Preguntas Incómodas?

¿Son los hombres más nobles que las mujeres? ¿Es verdad que las amistades que se mantienen durante toda la vida son las de la universidad? ¿Qué tendrá de cierto eso de que el primer amor nunca se olvida? ¿Es verdad que los amigos se lo cuentan todo? La mejor manera de disipar estas dudas es… ¡Dejar atrás todos estos tópicos y lanzarse a la piscina con **Preguntas Incómodas**!

Para ayudarte con esto nace **Preguntas Incómodas**, un juego que tendrás disponible para Android y iOS que permite saber cuánto conoces a tus amigos y qué saben ellos de ti, mediante preguntas "algo comprometidas".

Preguntas Incómodas, pretende ser el appgame de la temporada, mediante el cual, podrás retar a quien tu quieras a través de una pregunta tras la cual cada uno deberá adivinar qué ha contestado el otro. ¡Cuantos más aciertos mayor afinidad en los rankings de tus amigos! Además, también podrás consultar los rankings de tus contactos ¿Quién los conoce mejor que tú? Podrás invitar a tus amigos por Facebook e incluso publicar las respuestas de los demás, ¿estás preparado para conocerlos más a fondo?



Si quieres saber más sobre el juego, te dejamos algunas de las **preguntas incómodas** que encontrarás en la app.

- "¿Te gusta algún compañero de tu trabajo?"
- "¿Has sido infiel?"
- "¿Alguna vez el alcohol te ha hecho vomitar?"
- "¿Tienes dislexia y lo no bases?"

¡Si te han parecido graciosas o quieres saber las respuestas no dudes en bajarte la app! Además, si tienes alguna **pregunta incómoda** que no está en la app pero te encantaría poder preguntársela a tus amigos puedes publicarla en Twitter con el hashtag #PreguntasIncomodas y el equipo de Manduka Games la añadiremos en el juego.

Signat:

Manel Simon i Martínez
18 de Juny de 2015

Preguntas Incómodas: Resum

CATALÀ

Preguntas Incómodas és un videojoc per a dispositius mòbils basat en la clàssica dinàmica de preguntes i respostes. Al jugar amb amics i coneguts, els usuaris poden descobrir els seus secrets i intimitats, ja que el joc consisteix en respondre preguntes de caire personal i íntim. Alimentant aquest desig de coneixença social i fomentant l'entreteniment, Preguntas Incómodas pretén ser un projecte rendible econòmicament gràcies al model de negoci de compres integrades a l'aplicació.

ESPAÑOL

Preguntas incómodas es un videojuego para dispositivos móviles basado en la clásica dinámica de preguntas y respuestas. Al jugar con amigos y conocidos, los usuarios pueden descubrir sus secretos e intimidades, ya que el juego consiste en responder preguntas de carácter personal e íntimo. Alimentando este deseo de conocimiento social y fomentando el entretenimiento, Preguntas incómodas pretende ser un proyecto rentable económicamente gracias al modelo de negocio de compras integradas en la aplicación.

ENGLISH

Preguntas Incómodas is a game for mobile devices based on the classical dynamics of questions and answers. Playing with friends and acquaintances, users can discover their secrets and intimacies, since the game involves answering questions both personal and intimate. Fueling this desire for knowledge and promoting social entertainment, Preguntas Incómodas aims to be an economically profitable project through the business model of in-app purchases.